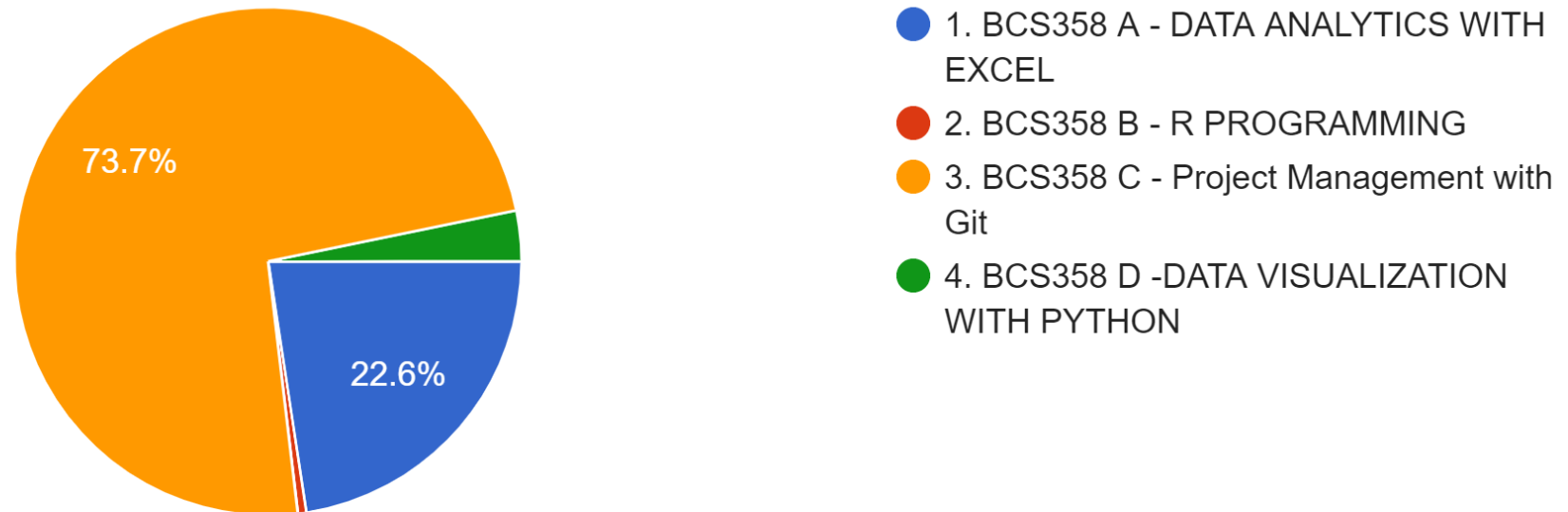


# SELECT AE COURSE – OPTIONS ARE

SELECT YOUR AE COURSE

186 responses



---

# BCS358C- Project Management with Git

Version Control with Git, 3rd Edition, by Prem Kumar Ponuthurai, Jon Loeliger Released October 2022,  
Publisher(s): O'Reilly Media, Inc.

- Pro Git book, written by Scott Chacon and Ben Straub and published by Apress, <https://git-scm.com/book/en/v2>
- [https://infyspringboard.onwingspan.com/web/en/app/toc/lex\\_auth\\_0130944433473699842782\\_shared/overview](https://infyspringboard.onwingspan.com/web/en/app/toc/lex_auth_0130944433473699842782_shared/overview)
- [https://infyspringboard.onwingspan.com/web/en/app/toc/lex\\_auth\\_01330134712177459211926\\_shared/overview](https://infyspringboard.onwingspan.com/web/en/app/toc/lex_auth_01330134712177459211926_shared/overview)  
1154..0099..22002233  
(As per VTU Syllabus)

# Program Outcomes:

## PROGRAM OUTCOMES

PO's	PO Description
P01	<b>Engineering knowledge:</b> Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
P02	<b>Problem analysis:</b> Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
P03	<b>Design/development of solutions:</b> Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
P04	<b>Conduct investigations of complex problems:</b> Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
P05	<b>Modern tool usage:</b> Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
P06	<b>The engineer and society:</b> Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
P07	<b>Environment and sustainability:</b> Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
P08	<b>Ethics:</b> Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

# Program Outcomes:

P09	<b>Individual and team work:</b> Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
P010	<b>Communication:</b> Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
P011	<b>Project management and finance:</b> Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
P012	<b>Life-long learning:</b> Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

# Program Specific Outcomes:

## PROGRAM SPECIFIC OUTCOMES

PSO's	PSO Description
<b>PSO1</b>	An ability to design and analyze algorithms by applying theoretical concepts to build complex and computer- based systems in the domain of System Software, Computer Networks & Security, Web technologies, Data Science and Analytics.
<b>PSO2</b>	Be able to develop various software solutions by applying the techniques of Data Base Management, Complex Mathematical Models, Software Engineering practices and Machine Learning with Artificial Intelligence.

# BCS358C-Project Management with Git

CREDITS – 1

CIE (Continuous Internal Evaluation) MARKS – 50

SEE (Semester End Examination) MARKS – 50

# BCS358C-Project Management with Git

## Course objectives(CO):-

- To familiar with basic command of Git
- To create and manage branches
- To understand how to collaborate and work with Remote Repositories
- To familiar with version controlling commands

# BCS358C-Project Management with Git

## Course Outcomes:

- Use the basics commands related to git repository
- Create and manage the branches
- Apply commands related to Collaboration and Remote Repositories
- Use the commands related to Git Tags, Releases and advanced git operations
- Analyse and change the git history



---

# BCS358C-Project Management with Git

---

# BCS358C-Project Management with Git

Sl.NO	Experiments
1	<b>Setting Up and Basic Commands</b>  Initialize a new Git repository in a directory. Create a new file and add it to the staging area and commit the changes with an appropriate commit message.
2	<b>Creating and Managing Branches</b>  Create a new branch named "feature-branch." Switch to the "master" branch. Merge the "feature-branch" into "master."
3	<b>Creating and Managing Branches</b>  Write the commands to stash your changes, switch  branches, and then apply the stashed changes.
4	<b>Collaboration and Remote Repositories</b>  Clone a remote Git repository to your local machine.
5	<b>Collaboration and Remote Repositories</b>  Fetch the latest changes from a remote repository and rebase your local branch onto the updated remote branch.
6	<b>Collaboration and Remote Repositories</b>  Write the command to merge "feature-branch" into "master" while providing a custom commit message for the merge.
7	<b>Git Tags and Releases</b>  Write the command to create a lightweight Git tag named "v1.0" for a commit in your local repository.
8	<b>Advanced Git Operations</b>

# BCS358C-Project Management with Git

	Write the command to cherry-pick a range of commits from "source-branch" to the current branch.
9	<b>Analysing and Changing Git History</b>  Given a commit ID, how would you use Git to view the details of that specific commit, including the author, date, and commit message?
10	<b>Analysing and Changing Git History</b>  Write the command to list all commits made by the author "JohnDoe" between "2023-01-01" and "2023-12-31."
11	<b>Analysing and Changing Git History</b>  Write the command to display the last five commits in the repository's history.
12	<b>Analysing and Changing Git History</b>  Write the command to undo the changes introduced by the commit with the ID "abc123".

# BCS358C-Project Management with Git

## What is Git?

- Git is a version control system which lets you track changes you make to your files over time.
- With Git, you can revert to various states of your files.
- You can also make a copy of your file, make changes to that copy, and then merge these changes to the original copy.

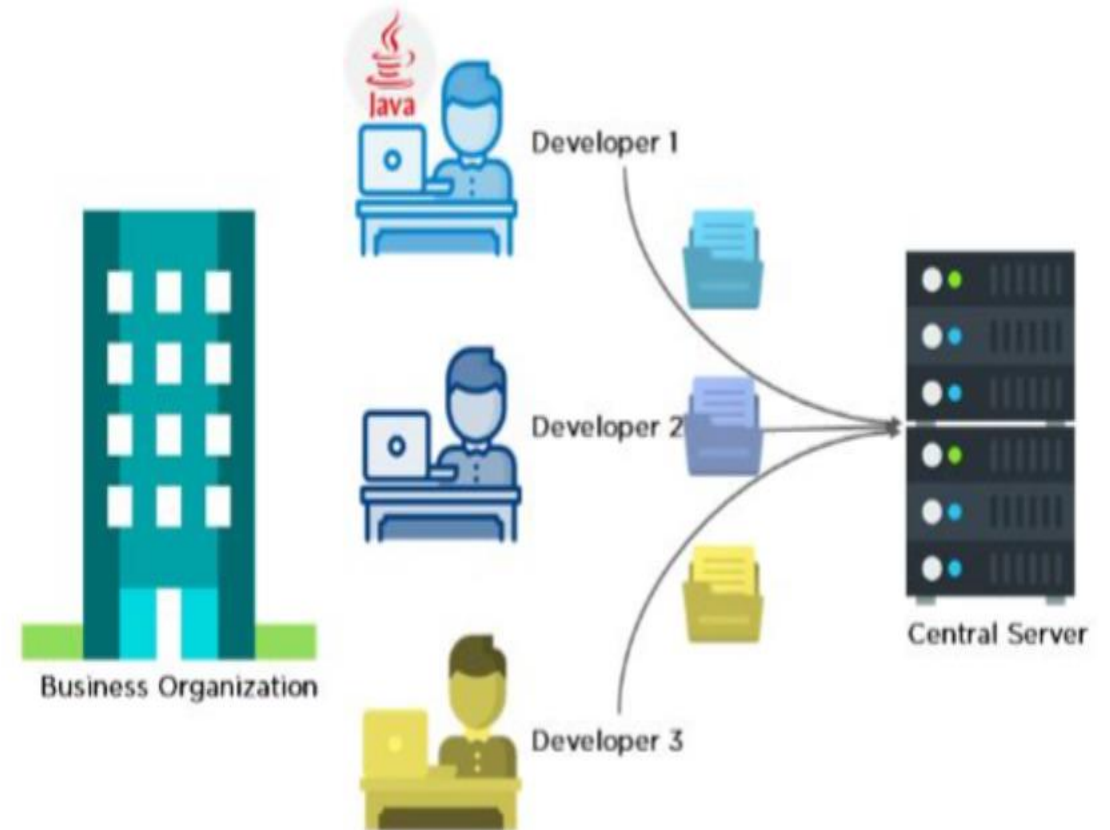
# Introduction to GIT

Git is a distributed version control system (VCS) that is widely used for tracking changes in source code during software development. It was created by Linus Torvalds in 2005 and has since become the de facto standard for version control in the software development industry.

# BCS358C-Project Management with Git

## Before going through Git

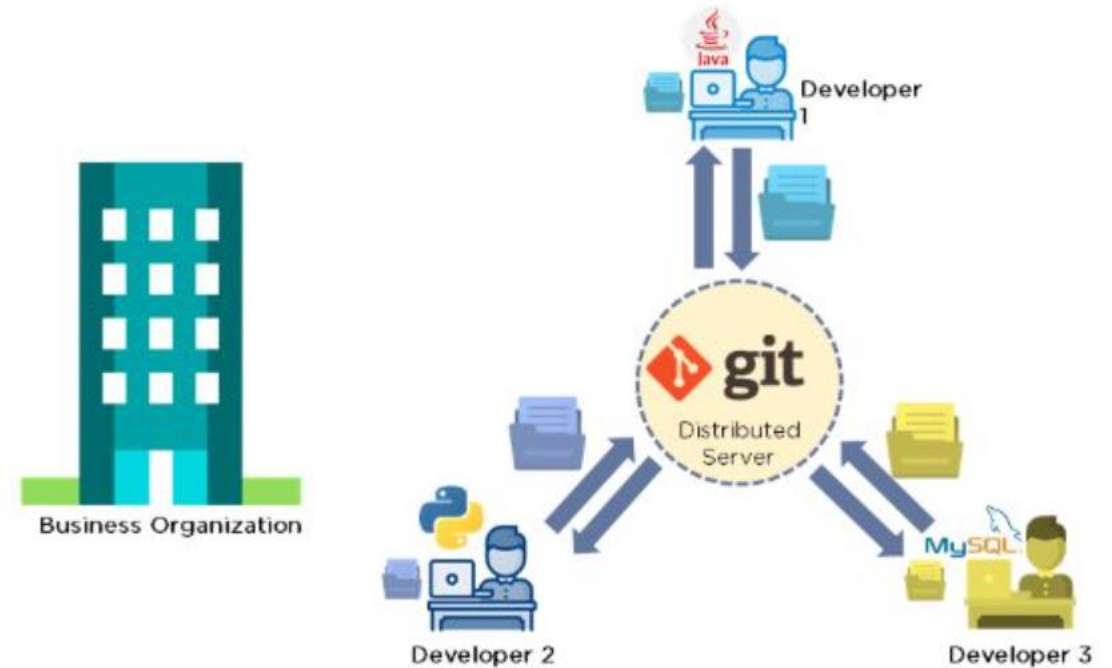
- Developers used to submit their codes to the central server without having copies of their own
- Any changes made to the source code were unknown to the other developers
- There was no communication between any of the developers



# BCS358C-Project Management with Git

## After Pass through the Git

- Every developer has an entire copy of the code on their local systems
- Any changes made to the source code can be tracked by others
- There is regular communication between the developers



# Introduction to GIT

- **Repository (Repo):** A Git repository is a directory or storage location where your project's files and version history are stored. There can be a local repository on your computer and remote repositories on servers.
- **Commits:** In Git, a commit is a snapshot of your project at a particular point in time. Each commit includes a unique identifier, a message describing the changes, and a reference to the previous commit.
- **Branches:** Branches in Git allow you to work on different features or parts of your project simultaneously without affecting the main development line (usually called the "master" branch). Branches make it easy to experiment, develop new features, and merge changes back into the main branch when they are ready.



# Introduction to GIT

- **Pull Requests (PRs):** In Git-based collaboration workflows, such as GitHub or GitLab, pull requests are a way for developers to propose changes and have them reviewed by their peers. This is a common practice for open-source and team-based projects.
- **Merging:** Merging involves combining changes from one branch (or multiple branches) into another. When a branch's changes are ready to be incorporated into the main branch, you can merge them.
- **Remote Repositories:** Remote repositories are copies of your project stored on a different server. Developers can collaborate by pushing their changes to a remote repository and pulling changes from it. Common remote repository hosting services include GitHub, GitLab, and Bitbucket.

# Introduction to GIT

- **Cloning:** Cloning is the process of creating a copy of a remote repository on your local machine. This allows you to work on the project and make changes locally.
- **Forking:** Forking is a way to create your copy of a repository, typically on a hosting platform like GitHub. You can make changes to your fork without affecting the original project and later create pull requests to contribute your changes back to the original repository.

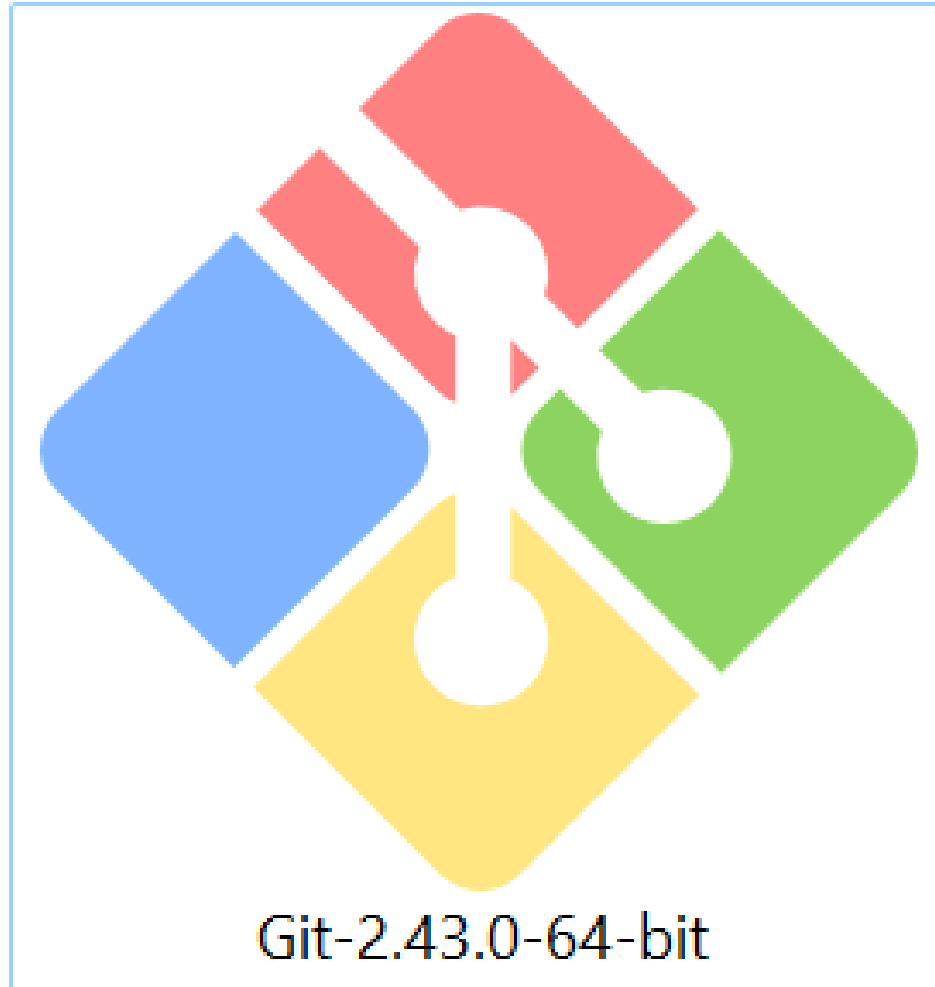
# Features to GIT

- ❑ **Version Control**
- ❑ **Collaboration**
- ❑ **Branching**
- ❑ **Distributed Development**
- ❑ **Backup and Recovery**
- ❑ **Code Review**
- ❑ **Open Source and Community Development**

## How to install Git

# BCS358C-Project Management with Git

---



# BCS358C-Project Management with Git

<https://git-scm.com/download/win>

# BCS358C-Project Management with Git

← → ↻ git-scm.com/download/win

**git** --distributed-is-the-new-centralized

Search entire site...

**About**

**Documentation**

**Downloads**

- GUI Clients
- Logos

**Community**

The entire **Pro Git book** written by Scott Chacon and Ben Straub is available to read [online for free](#). Dead tree versions are available on [Amazon.com](#).

## Download for Windows

[Click here to download](#) the latest (**2.43.0**) **64-bit** version of **Git for Windows**. This is the most recent [maintained build](#). It was released **16 days ago**, on 2023-11-20.

### Other Git for Windows downloads

**Standalone Installer**

- [32-bit Git for Windows Setup.](#)
- [64-bit Git for Windows Setup.](#)

**Portable ("thumbdrive edition")**

- [32-bit Git for Windows Portable.](#)
- [64-bit Git for Windows Portable.](#)

**Using winget tool**

Install [winget tool](#) if you don't already have it, then type this command in command prompt or Powershell.

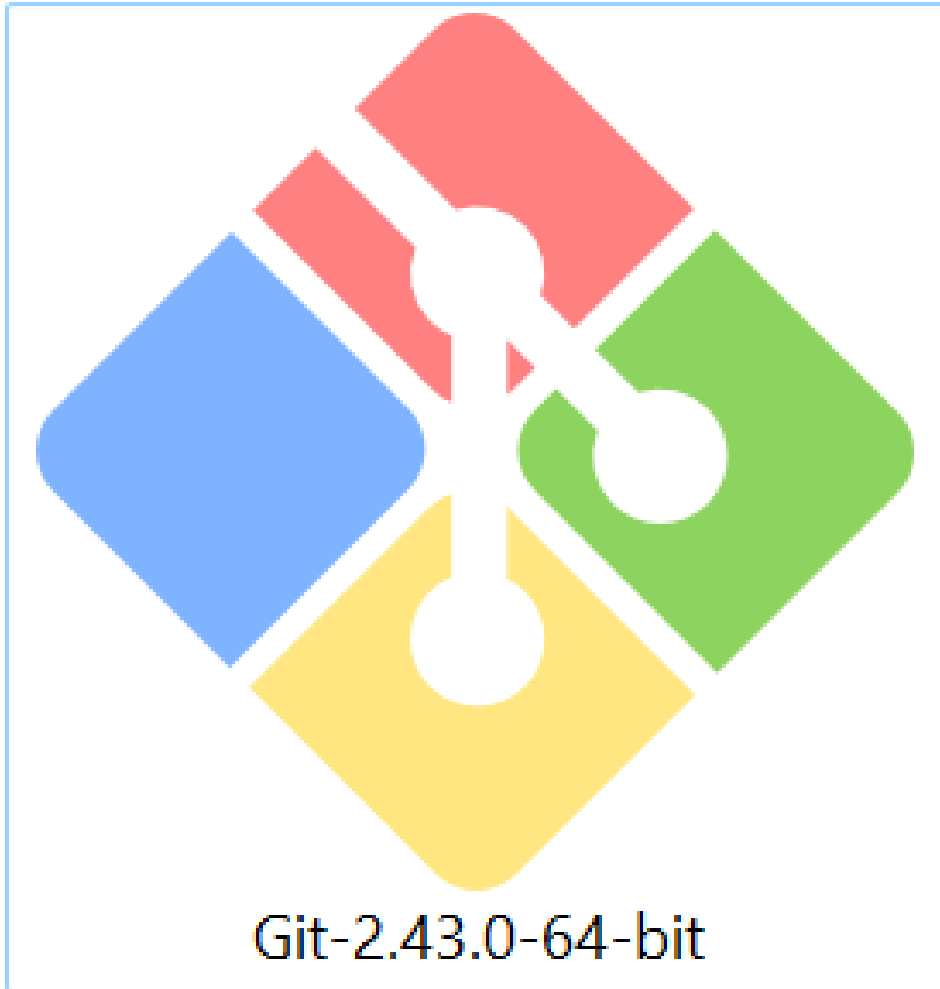
```
winget install --id Git.Git -e --source winget
```

The current source code release is version **2.43.0**. If you want the newer version, you can build it from [the](#)

Click and download

# BCS358C-Project Management with Git

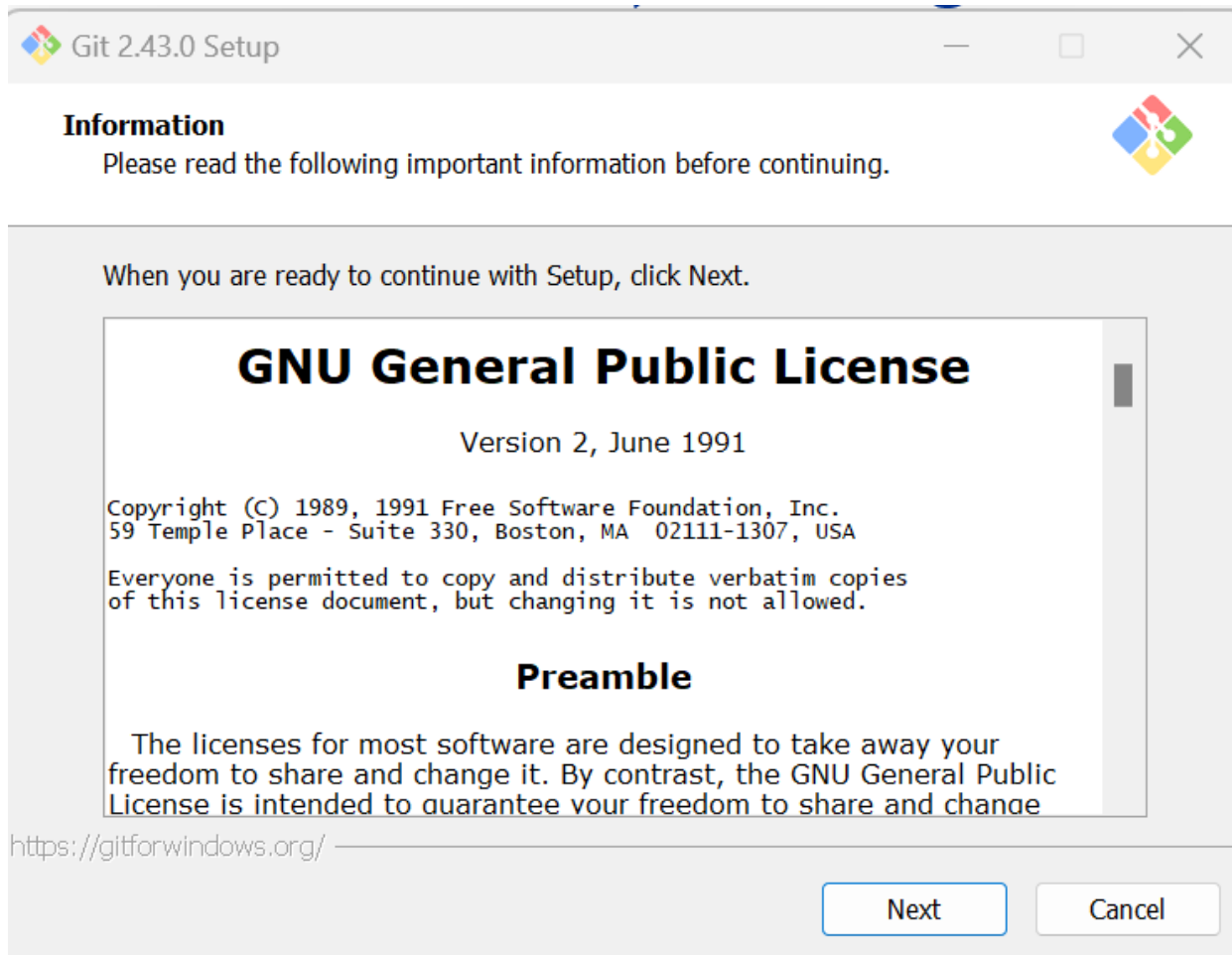
---



Check in your downloads  
Right Click  
And  
Select  
“Run As administrator”

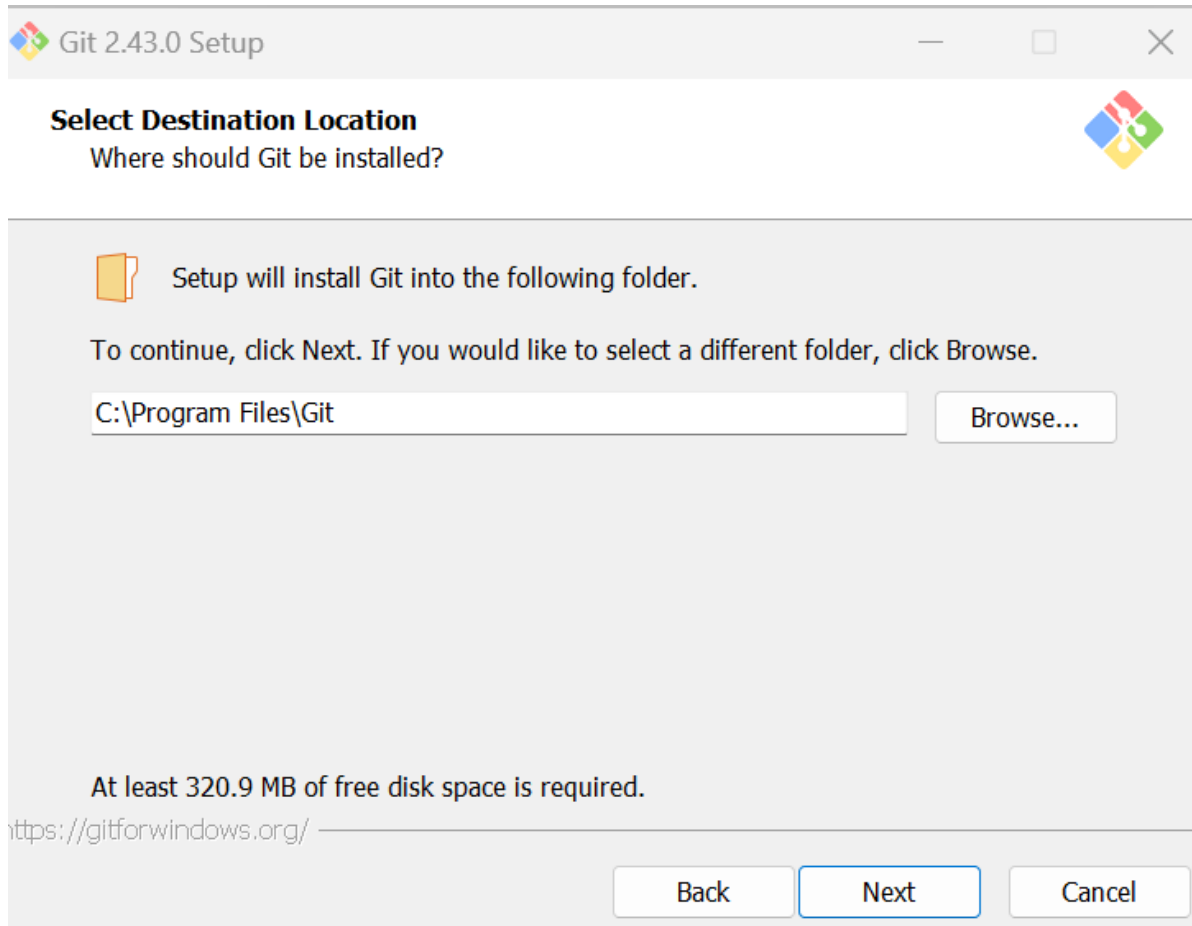


# BCS358C-Project Management with Git



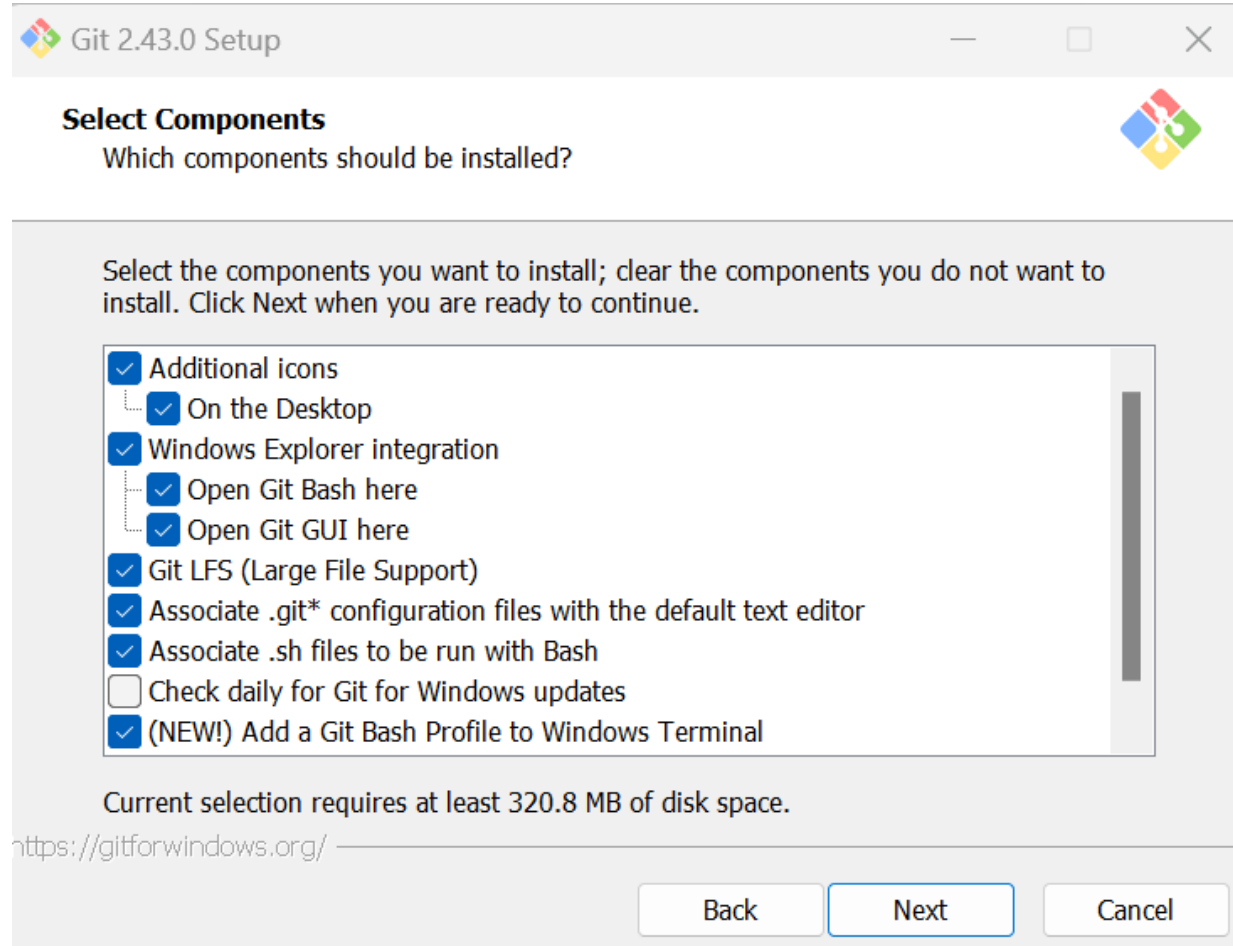
Click  
Next

# BCS358C-Project Management with Git



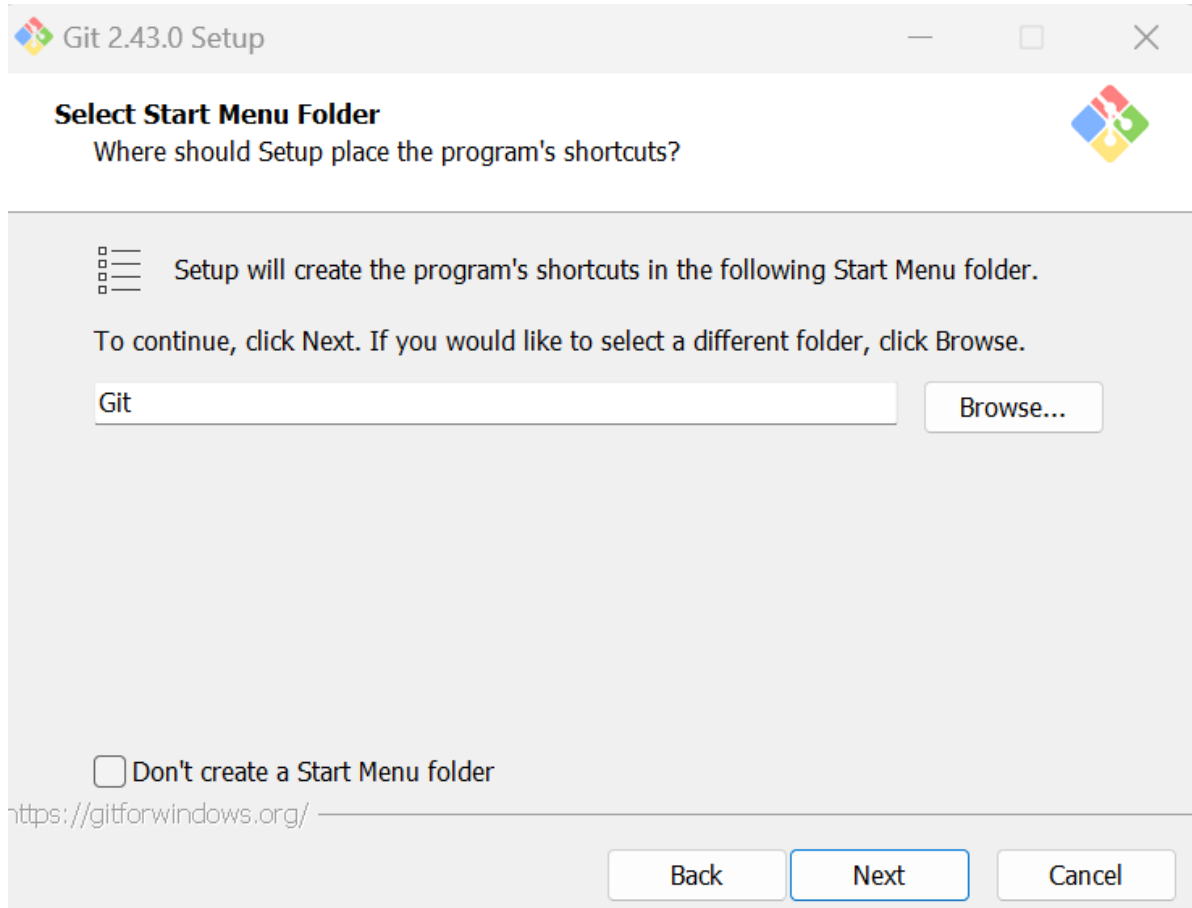
Click  
Next

# BCS358C-Project Management with Git



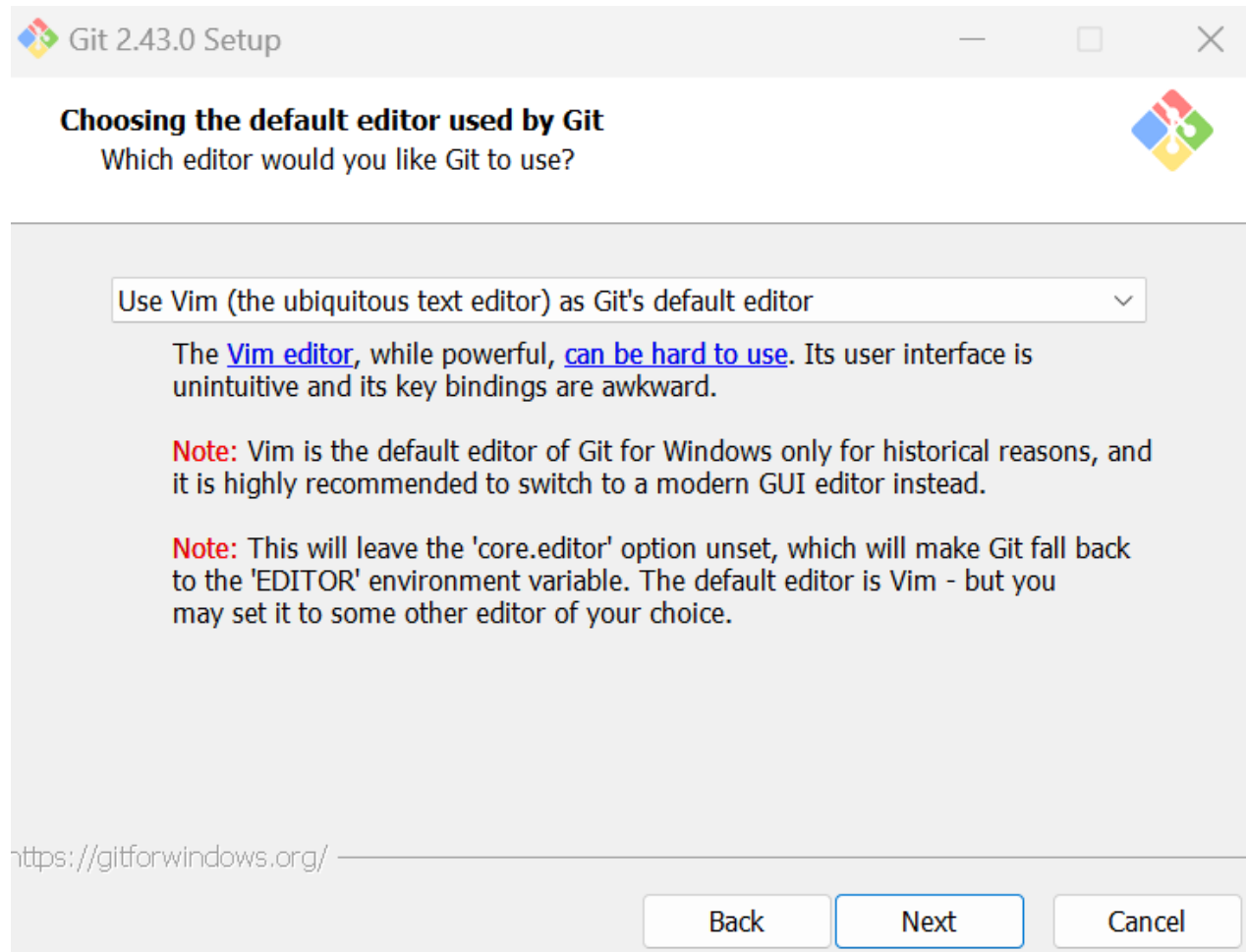
Select Additional  
icons &  
Click on  
Next

# BCS358C-Project Management with Git



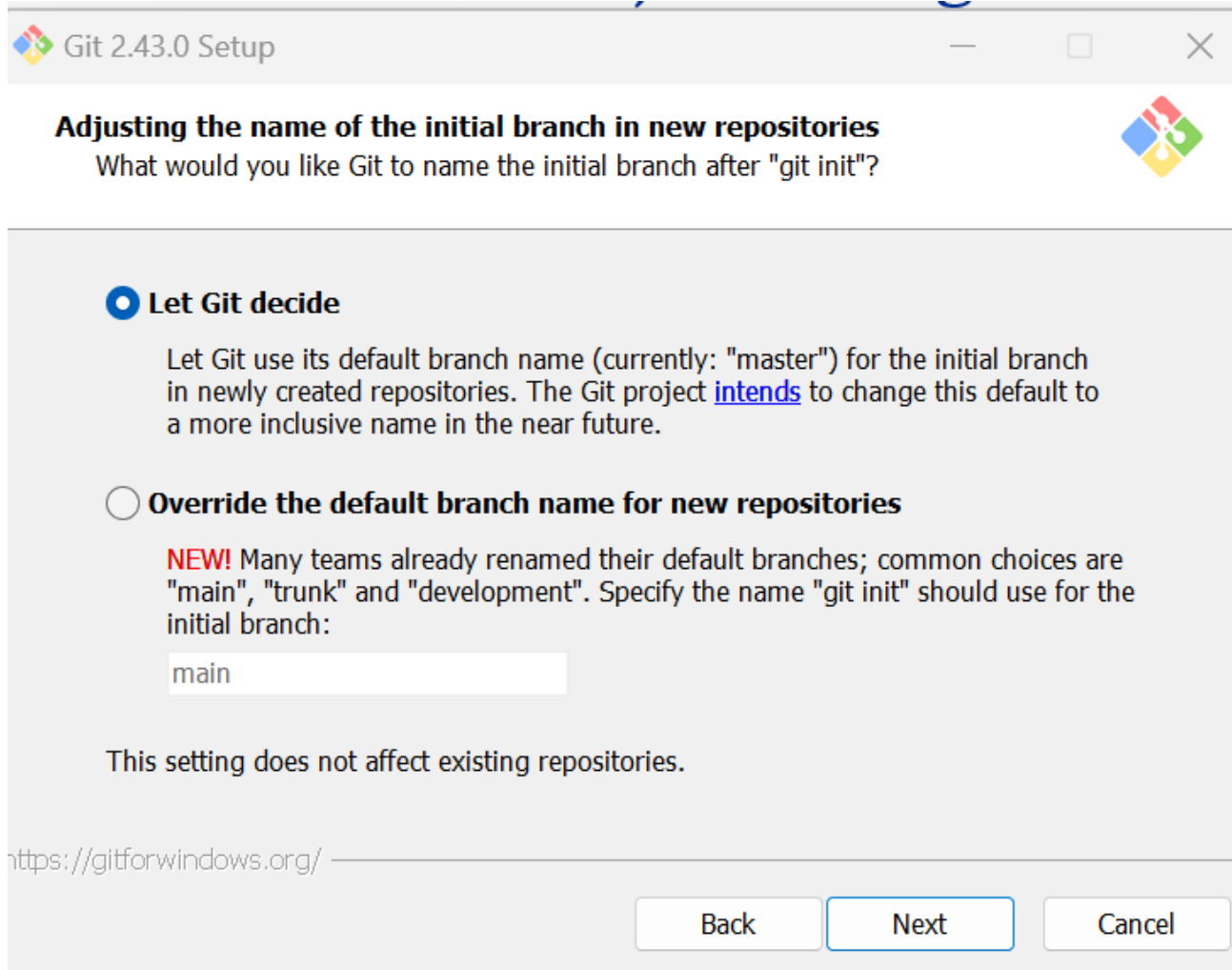
Click on  
Next

# BCS358C-Project Management with Git



Click on  
Next

# BCS358C-Project Management with Git



Git 2.43.0 Setup

**Adjusting the name of the initial branch in new repositories**  
What would you like Git to name the initial branch after "git init"?

☒ **Let Git decide**  
Let Git use its default branch name (currently: "master") for the initial branch in newly created repositories. The Git project [intends](#) to change this default to a more inclusive name in the near future.

☐ **Override the default branch name for new repositories**  
**NEW!** Many teams already renamed their default branches; common choices are "main", "trunk" and "development". Specify the name "git init" should use for the initial branch:

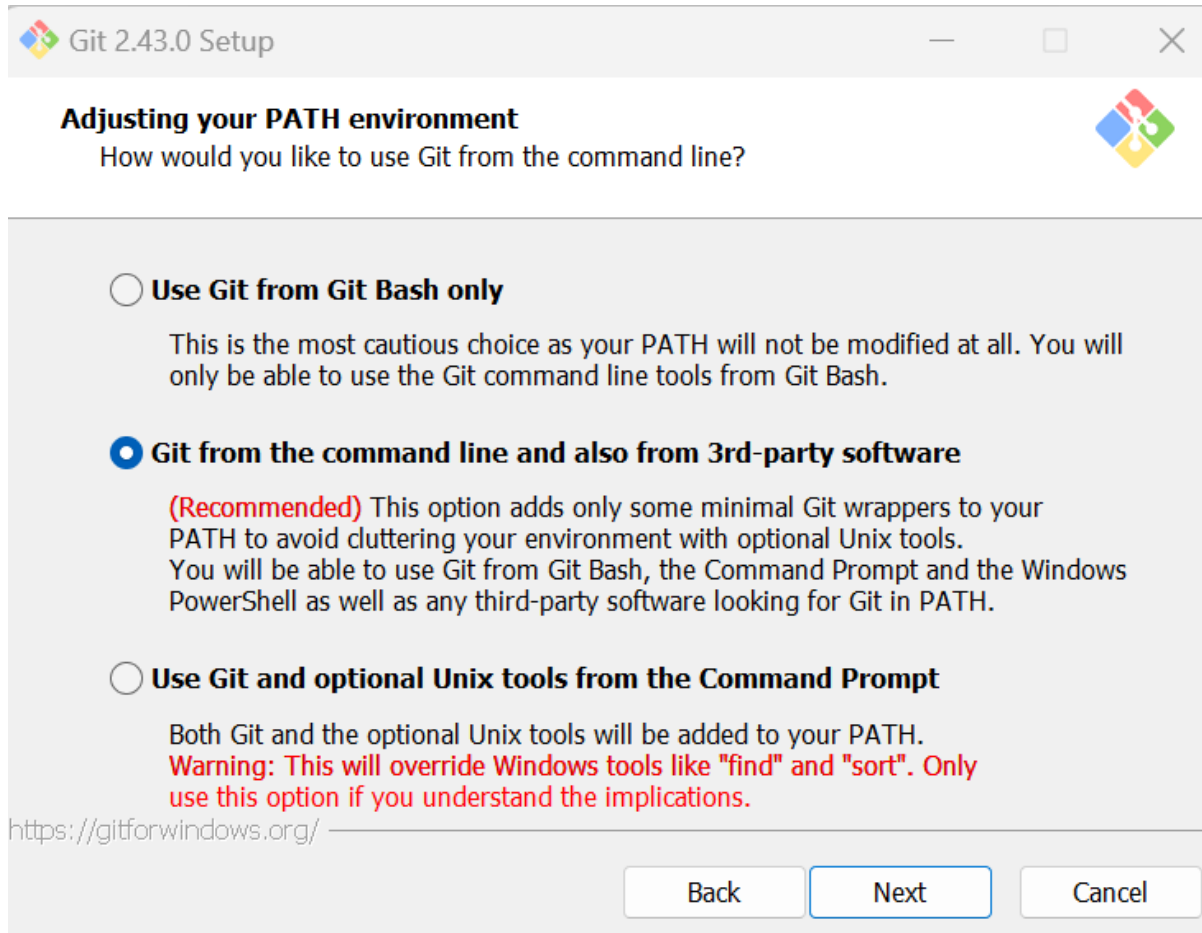
This setting does not affect existing repositories.

<https://gitforwindows.org/>

Back Next Cancel

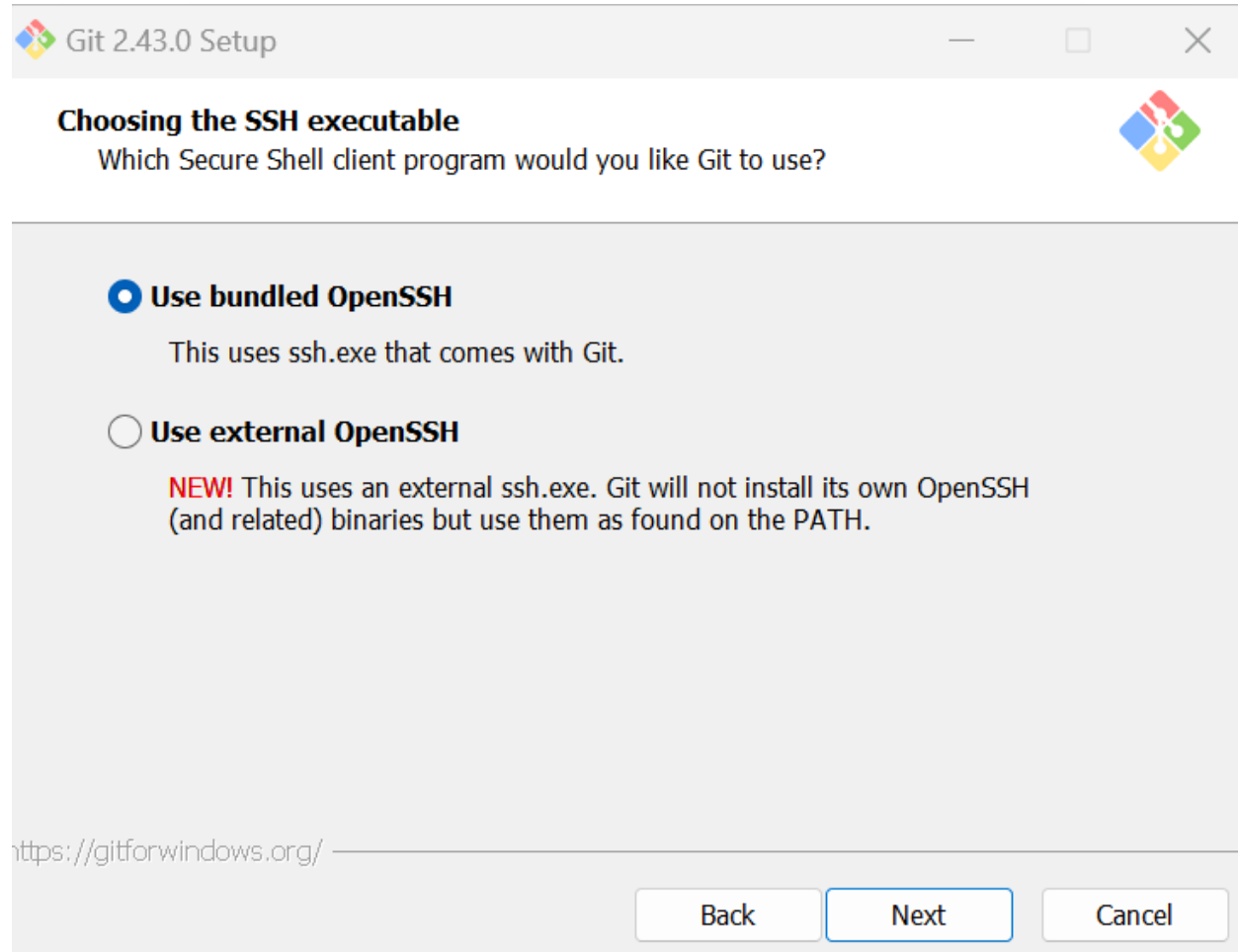
Click on  
Next

# BCS358C-Project Management with Git



Click on  
Next

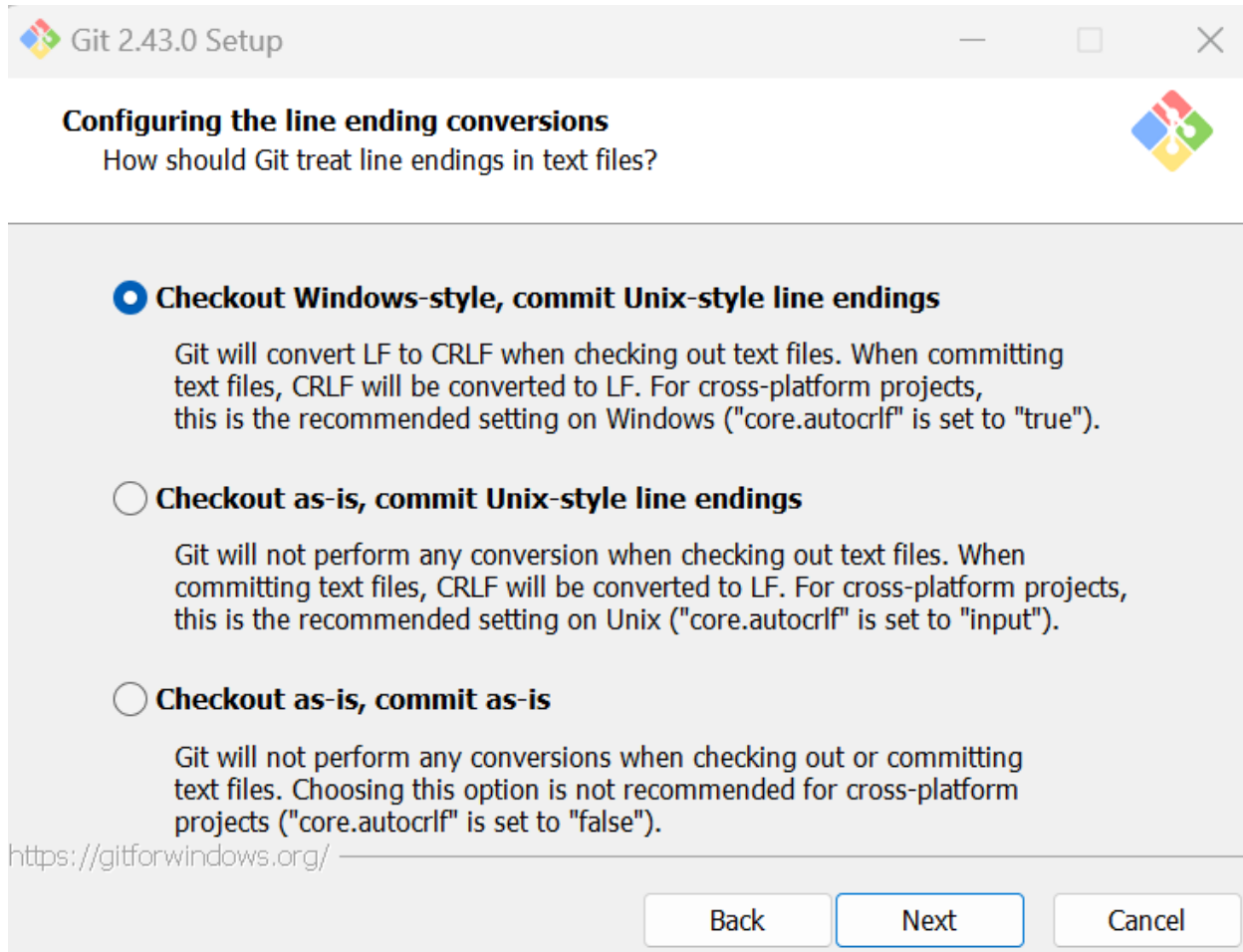
# BCS358C-Project Management with Git



Click on  
Next

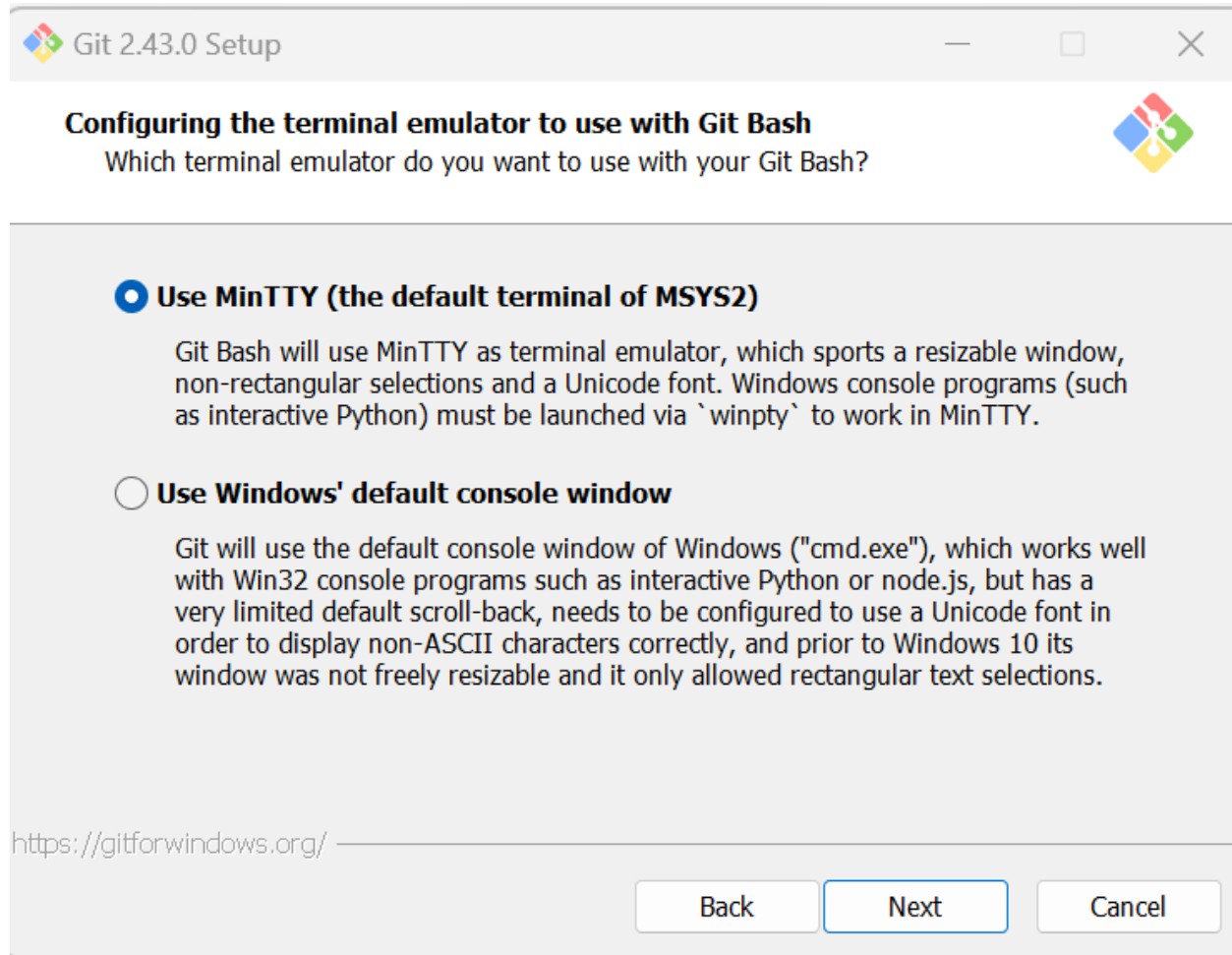


# BCS358C-Project Management with Git



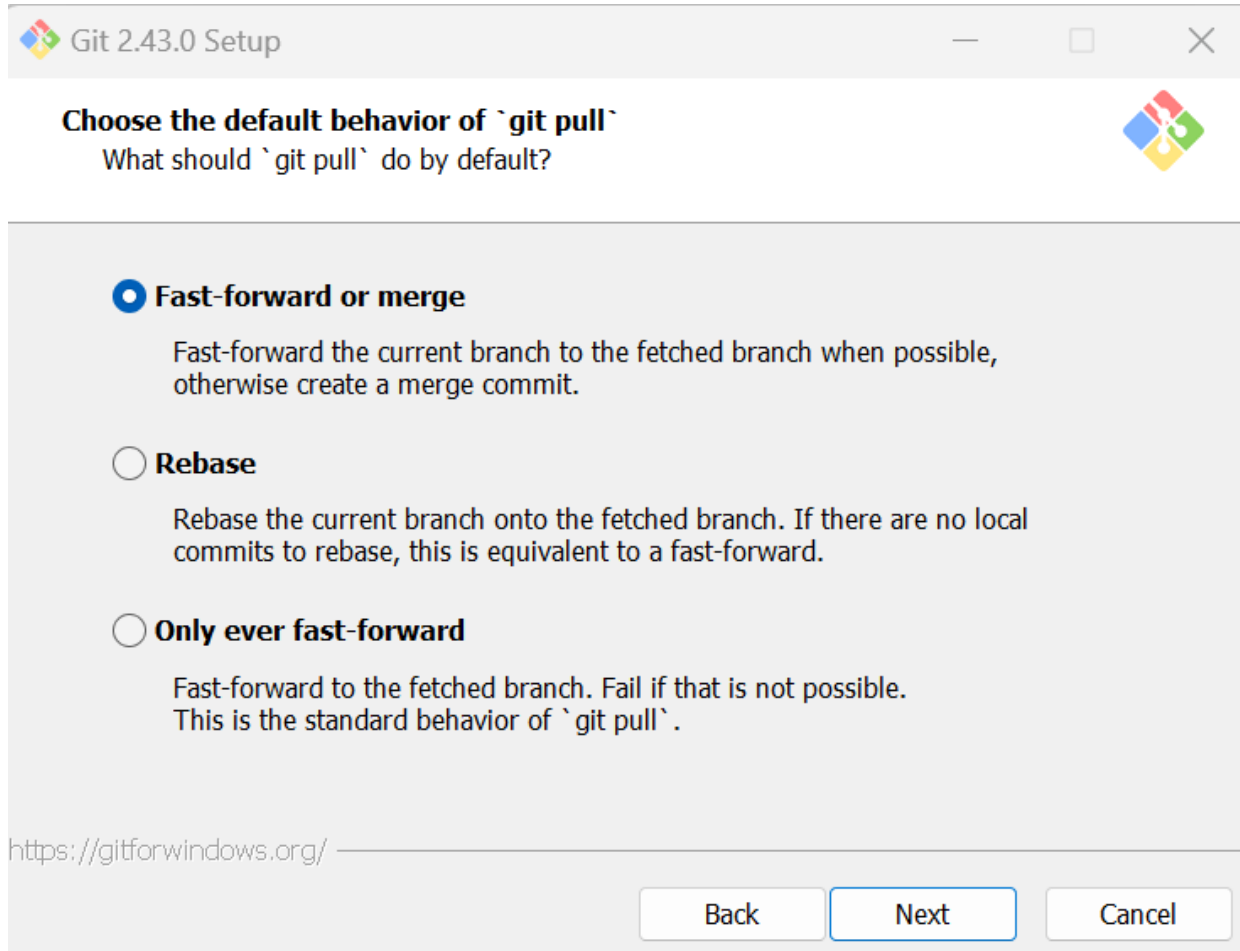
Click on  
Next

# BCS358C-Project Management with Git



Click on  
Next

# BCS358C-Project Management with Git



Git 2.43.0 Setup

Choose the default behavior of ``git pull``  
What should ``git pull`` do by default?

☒ **Fast-forward or merge**  
Fast-forward the current branch to the fetched branch when possible, otherwise create a merge commit.

☐ **Rebase**  
Rebase the current branch onto the fetched branch. If there are no local commits to rebase, this is equivalent to a fast-forward.

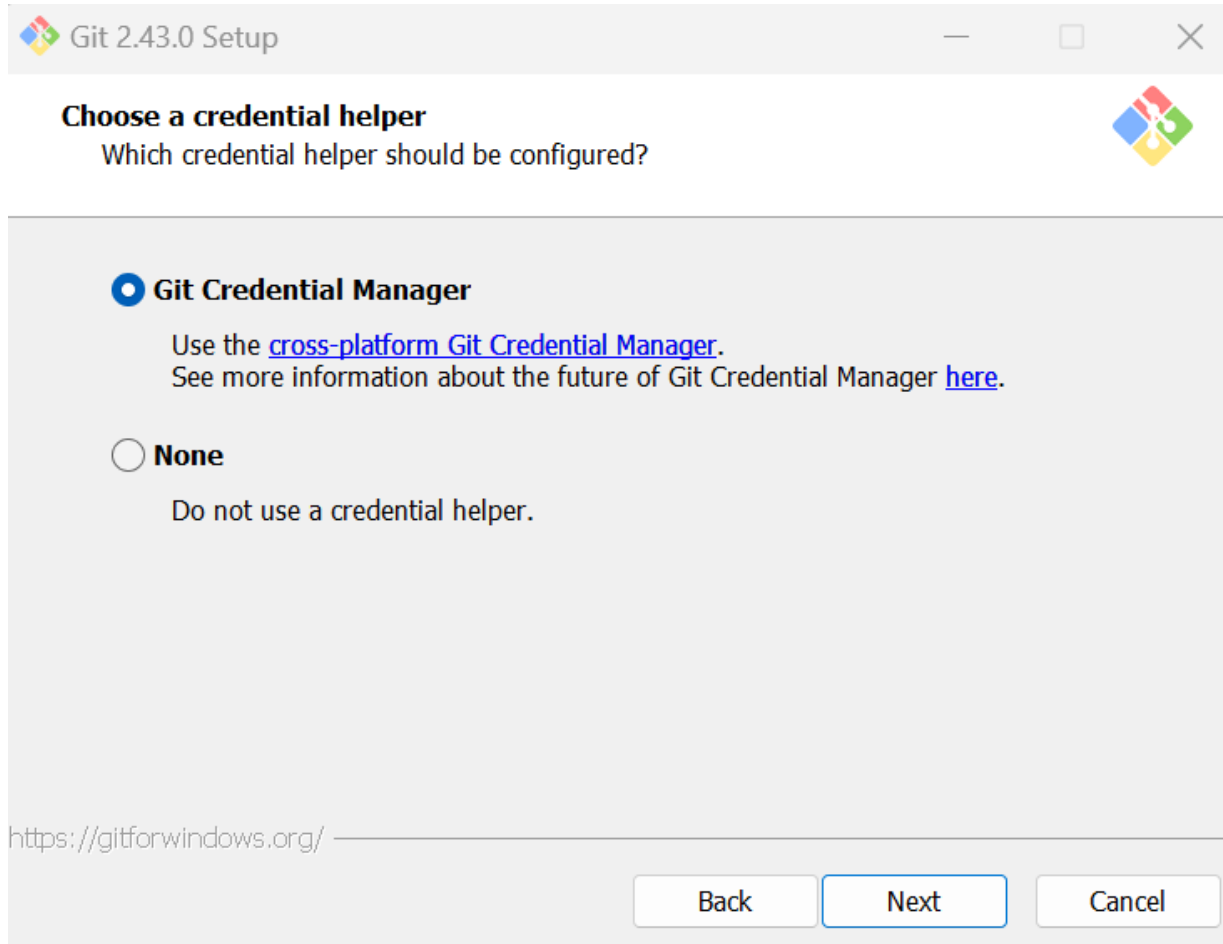
☐ **Only ever fast-forward**  
Fast-forward to the fetched branch. Fail if that is not possible. This is the standard behavior of ``git pull``.

<https://gitforwindows.org/>

Back Next Cancel

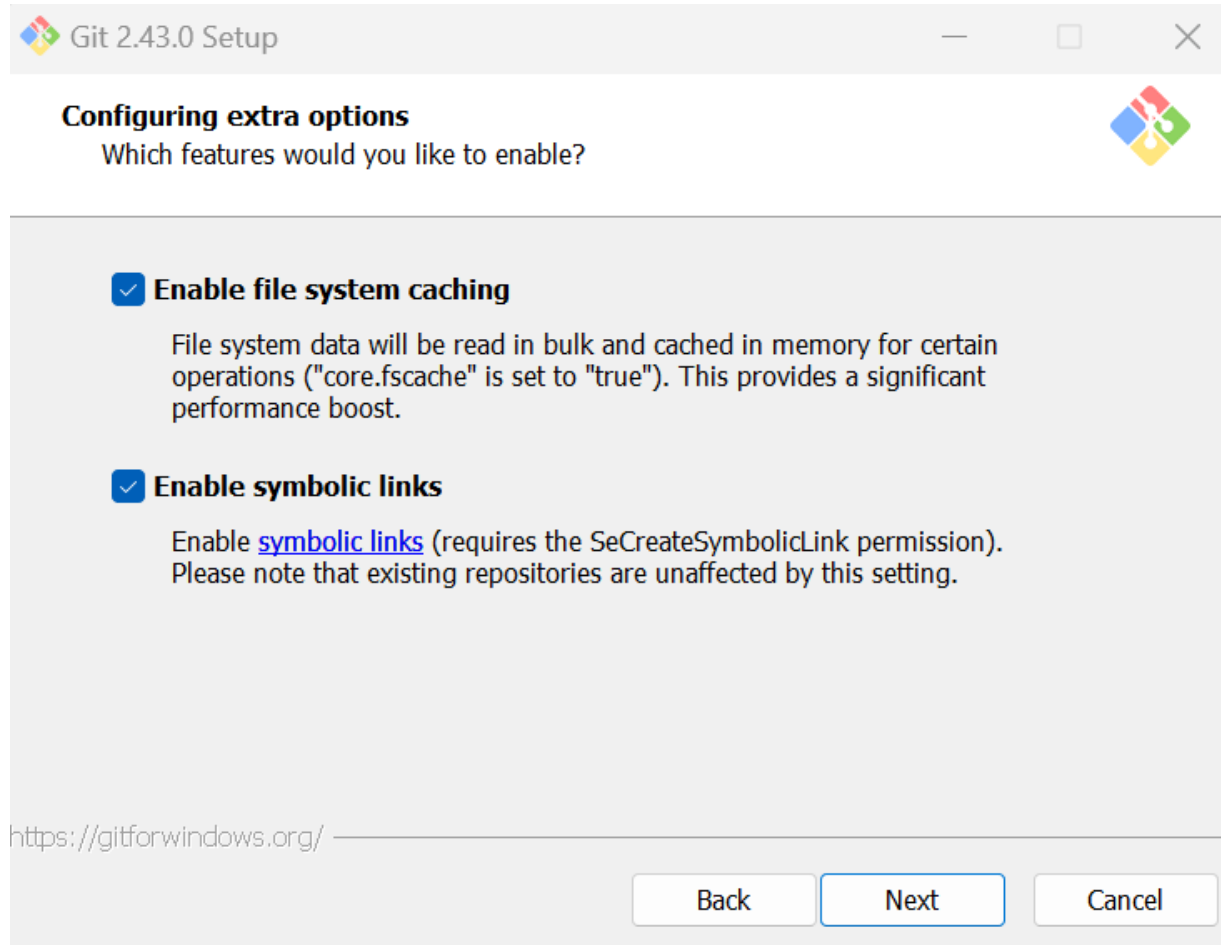
Click on  
Next

# BCS358C-Project Management with Git



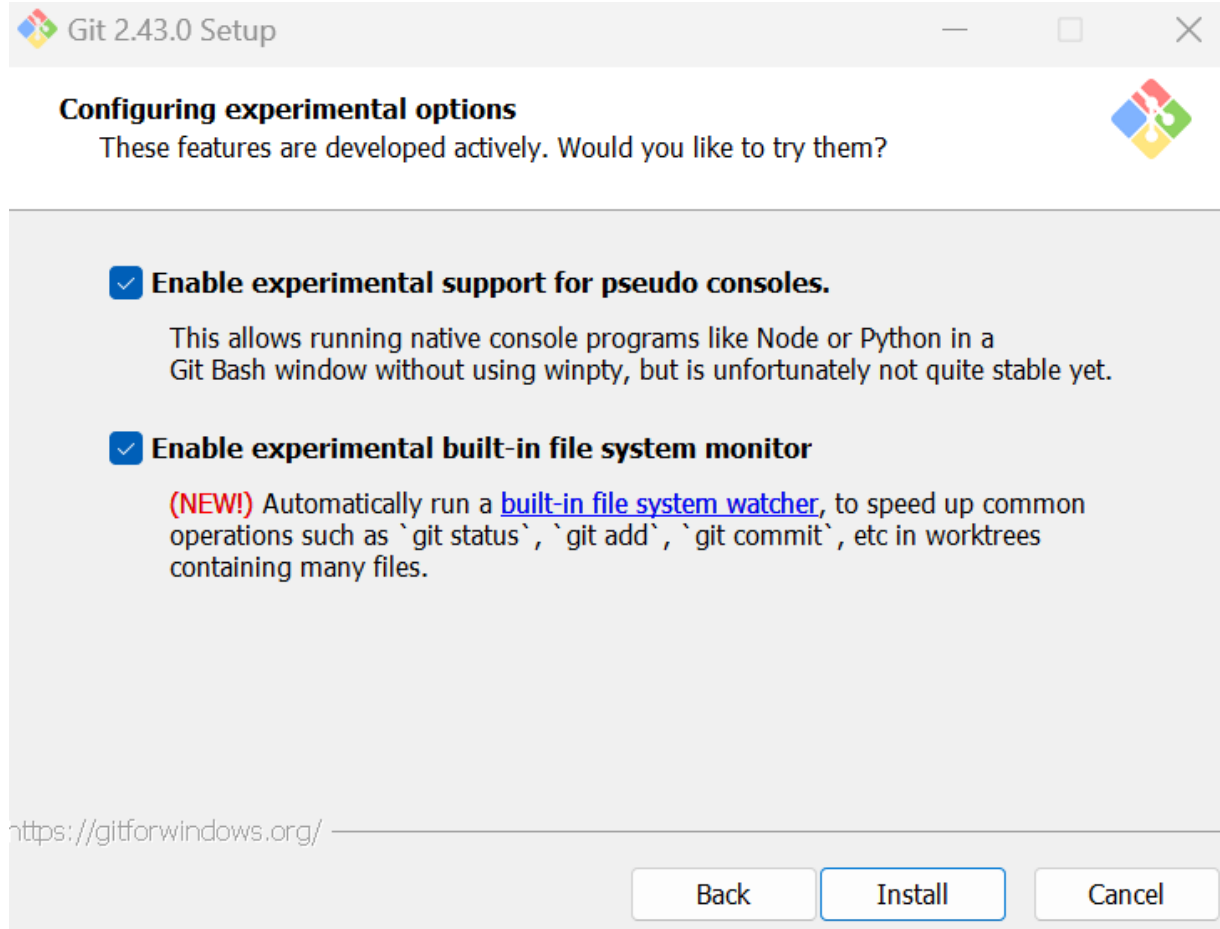
Click on  
Next

# BCS358C-Project Management with Git



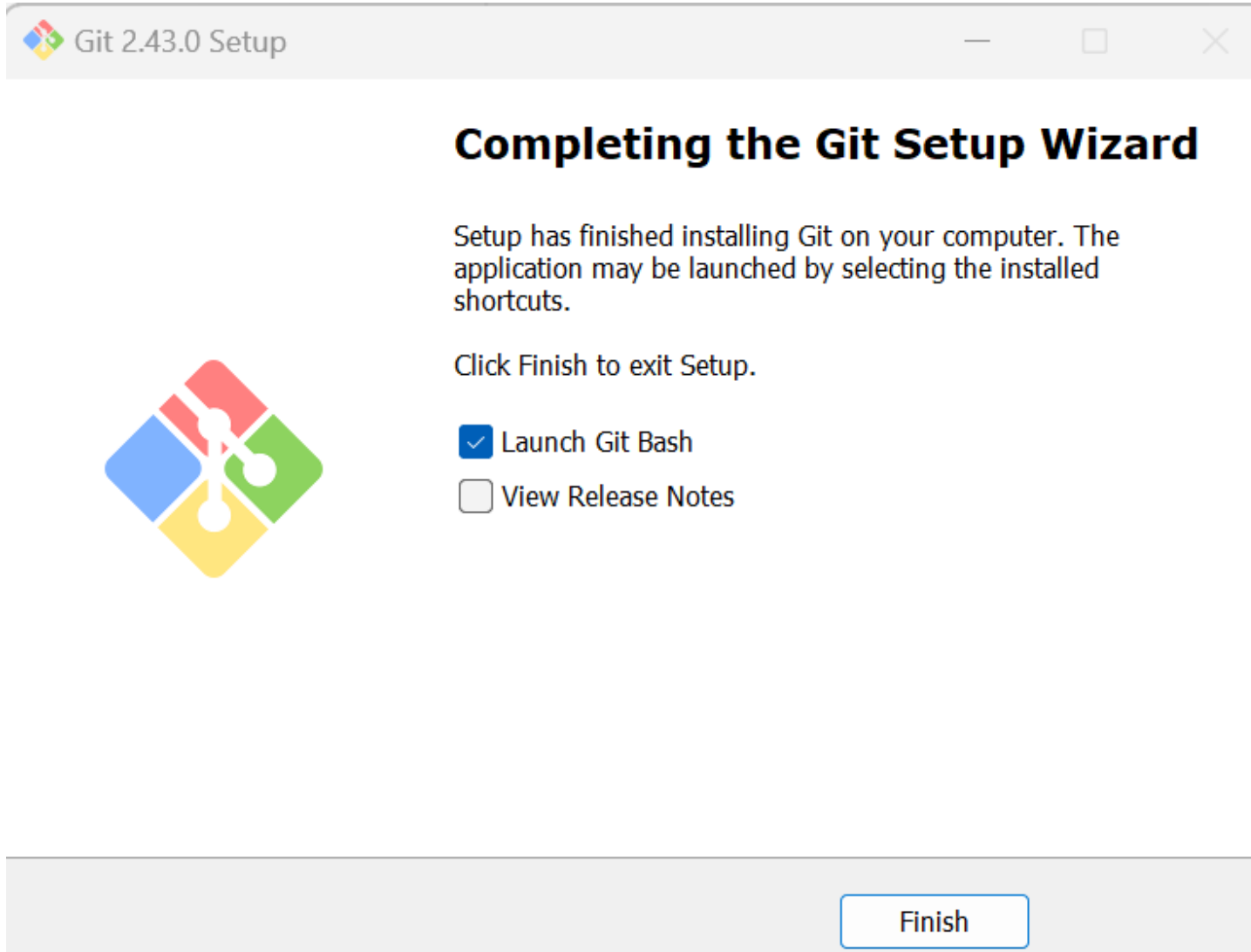
Click on  
Next

# BCS358C-Project Management with Git



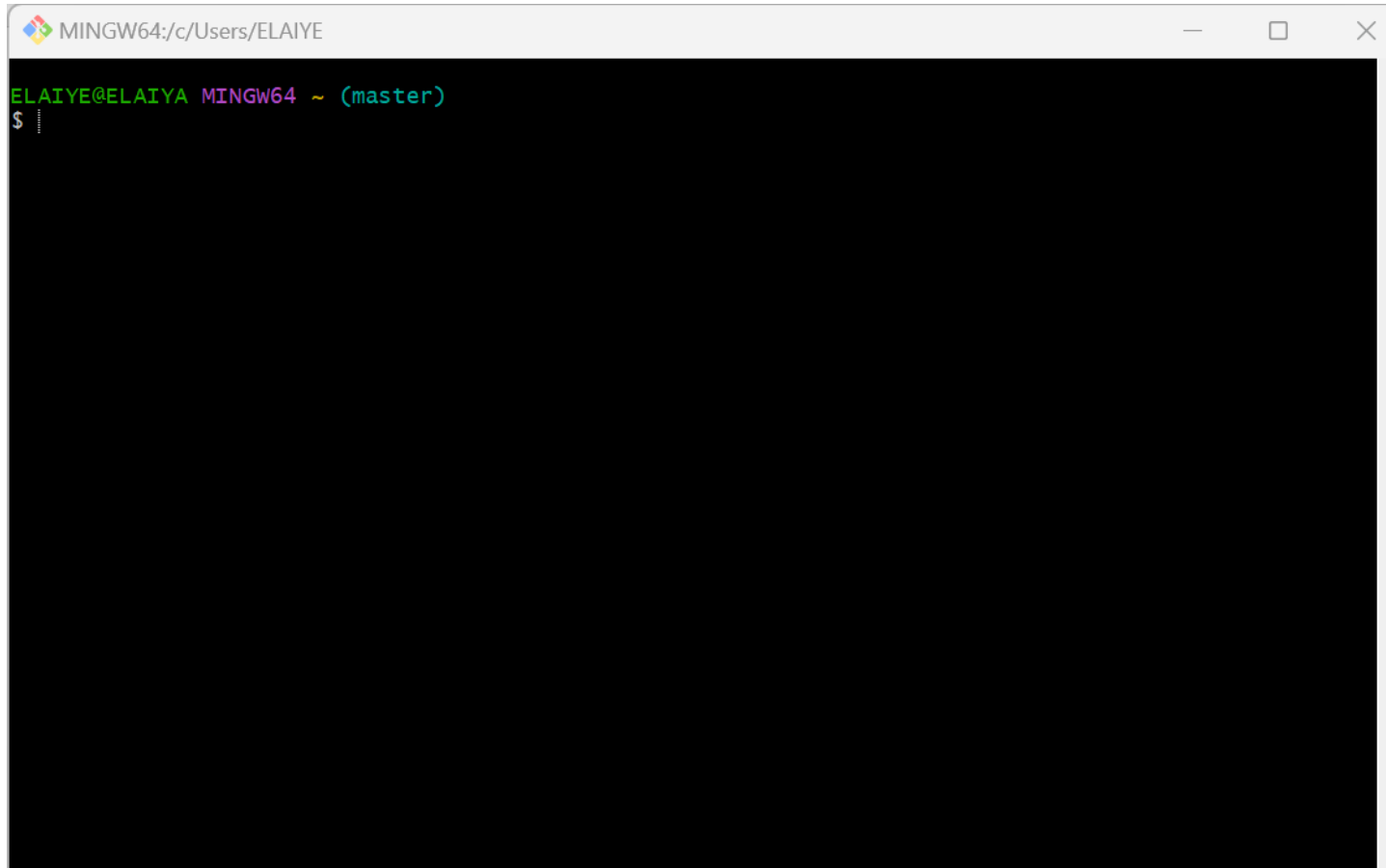
Click on  
Install

# BCS358C-Project Management with Git



Click on  
Finish

# BCS358C-Project Management with Git

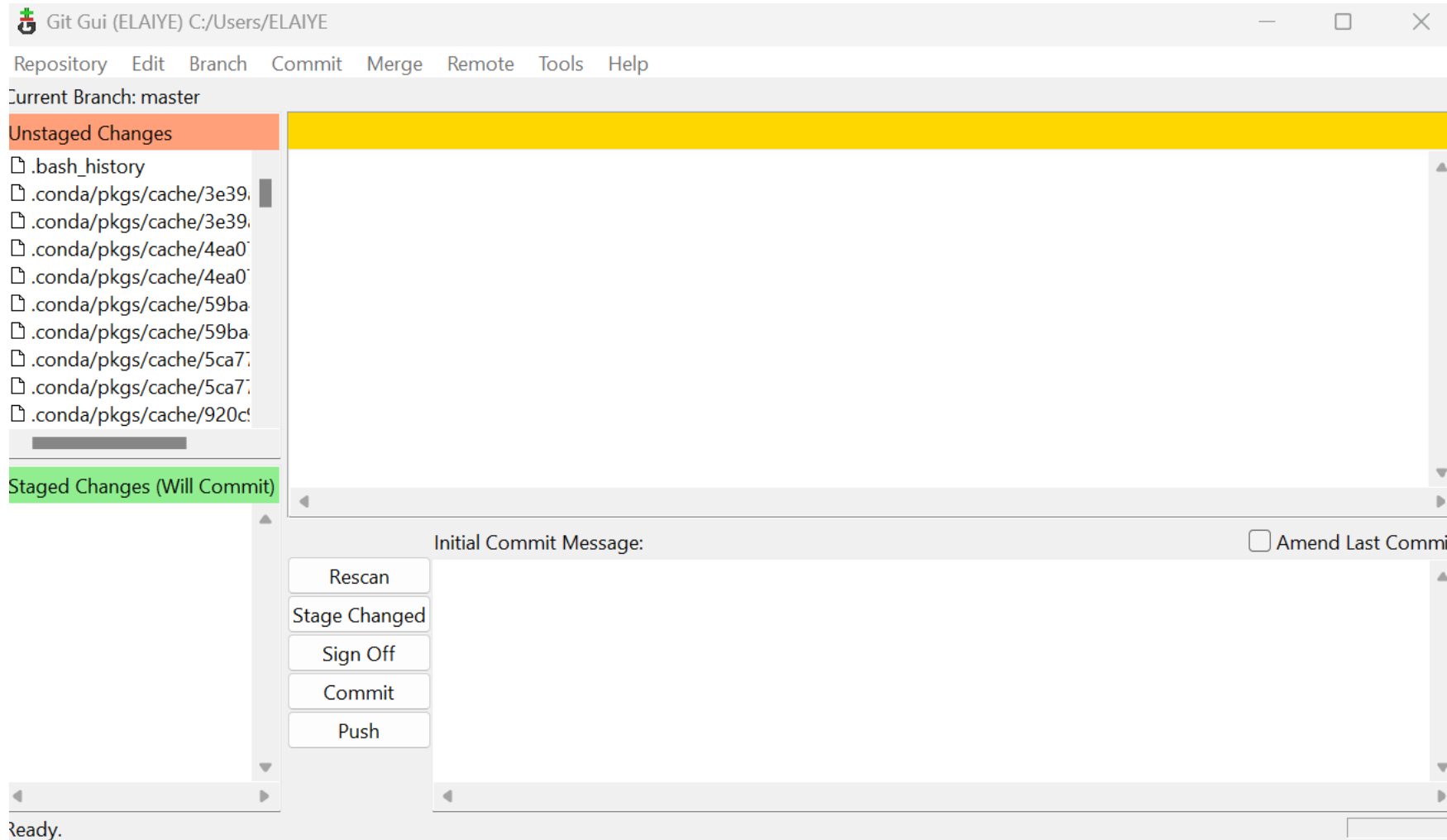


```
MINGW64:/c/Users/ELAIYE
ELAIYE@ELAIYA MINGW64 ~ (master)
$
```

Then you can view git bash Terminal



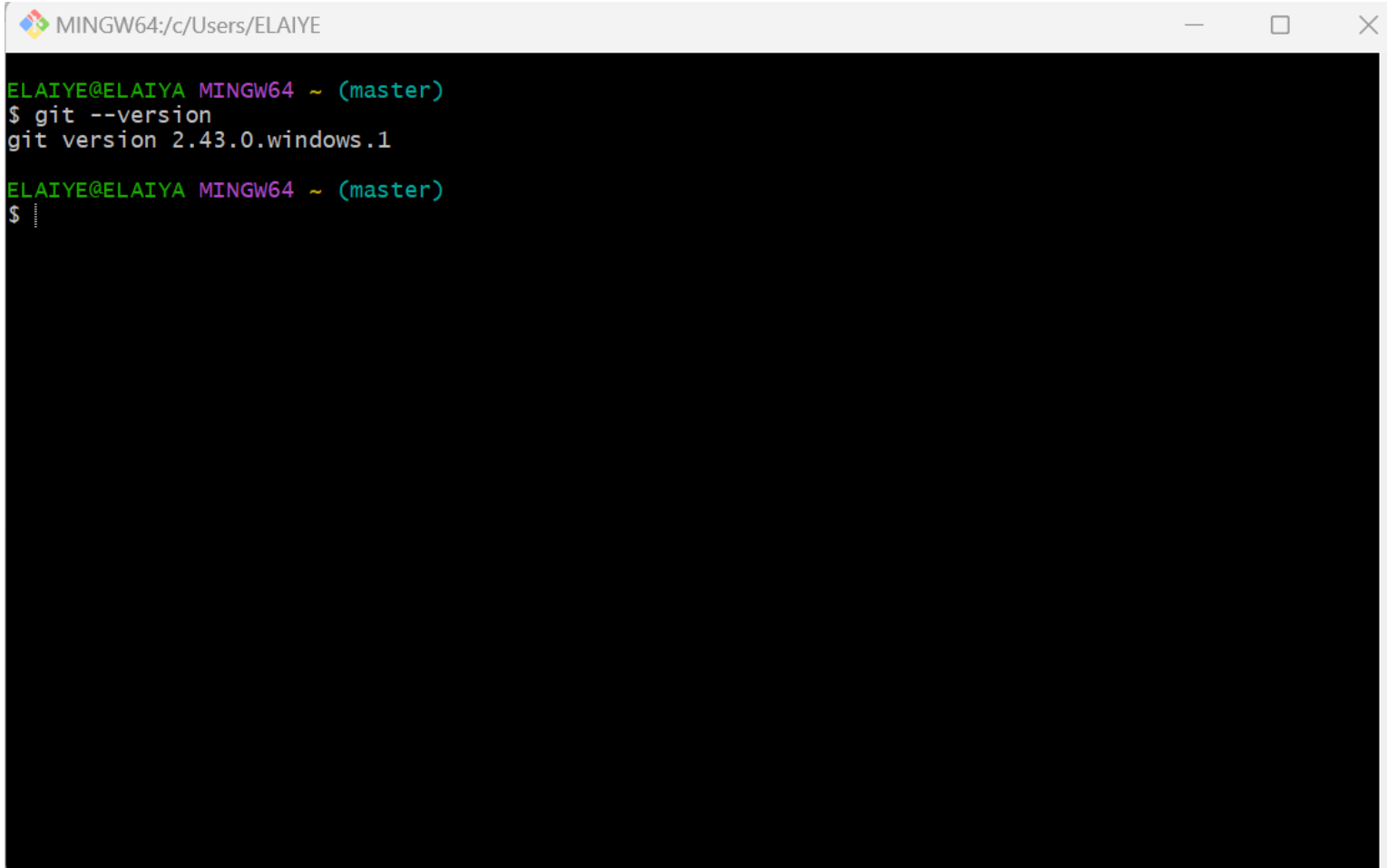
# BCS358C-Project Management with Git



Then you can view git GUI

## How to configure Git

# BCS358C-Project Management with Git

A screenshot of a Windows terminal window titled 'MINGW64:/c/Users/ELAIYE'. The prompt is 'ELAIYE@ELAIYA MINGW64 ~ (master)'. The user enters '\$ git --version' and the output is 'git version 2.43.0.windows.1'. The prompt is then '\$ ' with a cursor.

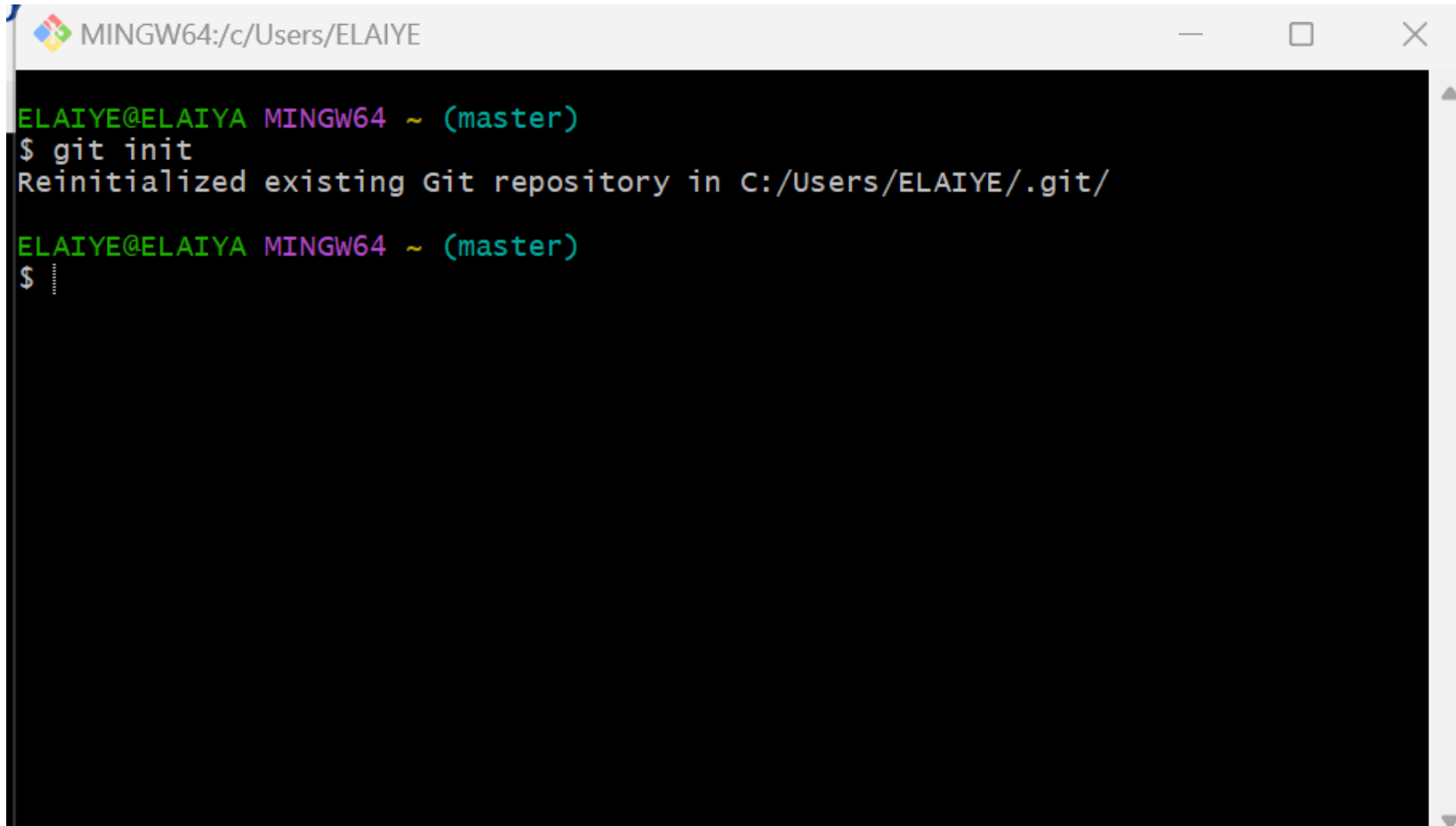
```
ELAIYE@ELAIYA MINGW64 ~ (master)
$ git --version
git version 2.43.0.windows.1
ELAIYE@ELAIYA MINGW64 ~ (master)
$
```

Type

- `git --version`

You can view the version which is installed in the system

# BCS358C-Project Management with Git

A screenshot of a Windows command prompt window titled 'MINGW64:/c/Users/ELAIYE'. The prompt shows the user 'ELAIYE@ELAIYA' in a 'MINGW64' environment at the '~' directory on the '(master)' branch. The user enters the command '\$ git init', and the output is 'Reinitialized existing Git repository in C:/Users/ELAIYE/.git/'. The prompt then shows the user entering '\$' again, with a cursor on the next line.

```
ELAIYE@ELAIYA MINGW64 ~ (master)
$ git init
Reinitialized existing Git repository in C:/Users/ELAIYE/.git/

ELAIYE@ELAIYA MINGW64 ~ (master)
$
```

Type

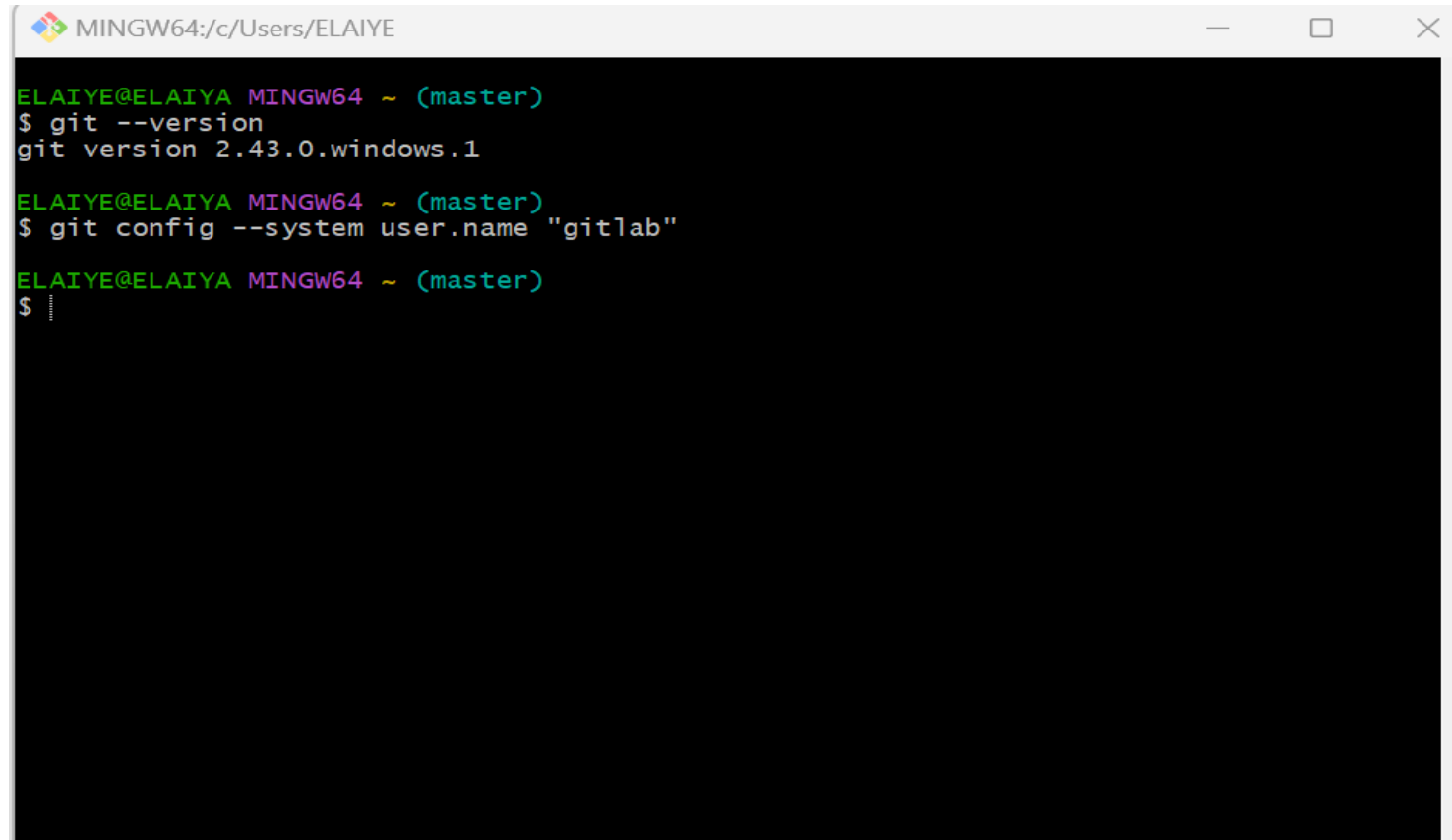
- **git init**

You can  
initialize git  
repository

## BCS358C-Project Management with Git

To set your username, type and execute these commands:

- `git config --system user.name "gitlab"`

A screenshot of a MINGW64 terminal window. The title bar shows the path 'MINGW64:/c:/Users/ELAIYE'. The terminal content shows the following sequence of commands and output:

```
ELAIYE@ELAIYA MINGW64 ~ (master)
$ git --version
git version 2.43.0.windows.1

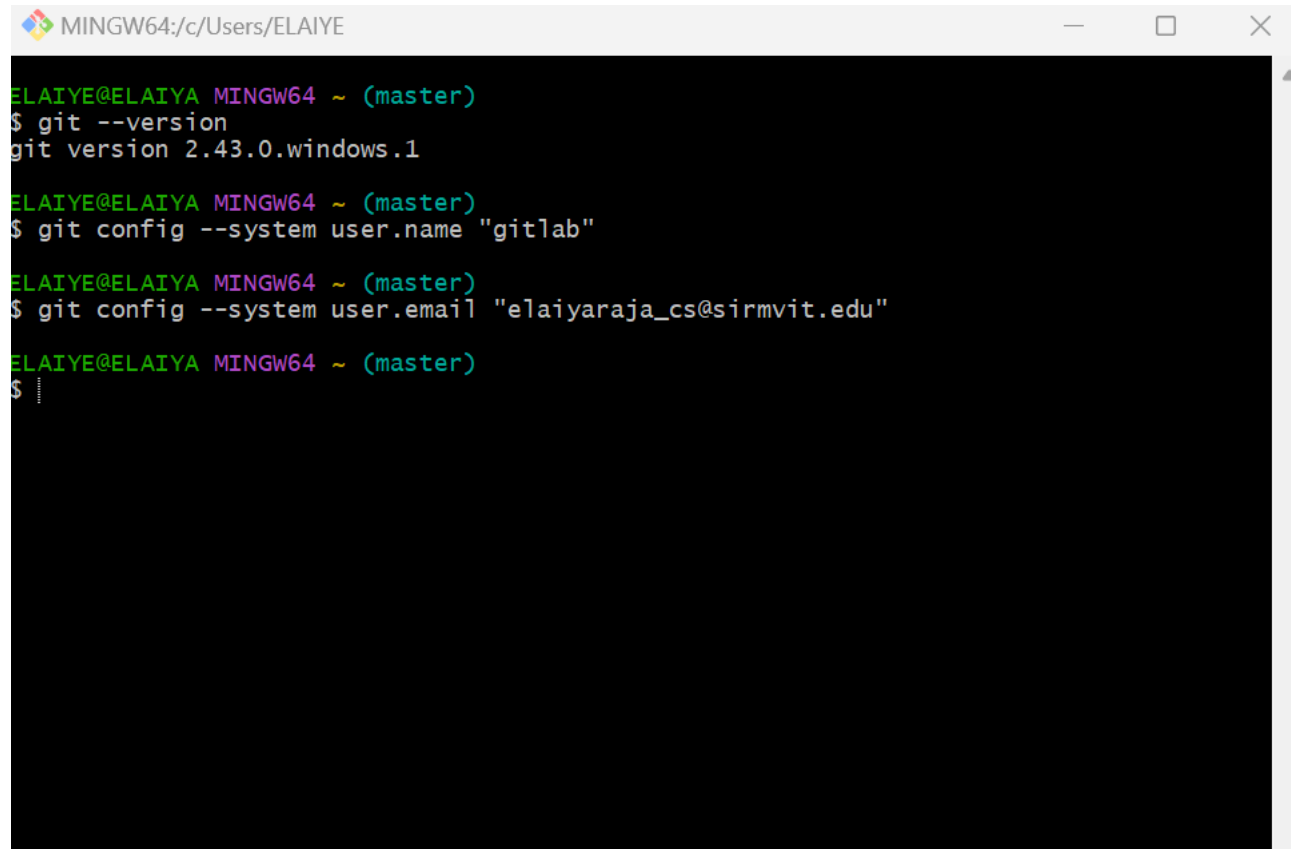
ELAIYE@ELAIYA MINGW64 ~ (master)
$ git config --system user.name "gitlab"

ELAIYE@ELAIYA MINGW64 ~ (master)
$
```

# BCS358C-Project Management with Git

To set your username, type and execute these commands:

- `git config --system user.email "elaiyaraja_cs@sirmvit.edu"`

A screenshot of a Windows terminal window titled 'MINGW64:/c/Users/ELAIYE'. The terminal shows the following commands and output:

```
ELAIYE@ELAIYA MINGW64 ~ (master)
$ git --version
git version 2.43.0.windows.1

ELAIYE@ELAIYA MINGW64 ~ (master)
$ git config --system user.name "gitlab"

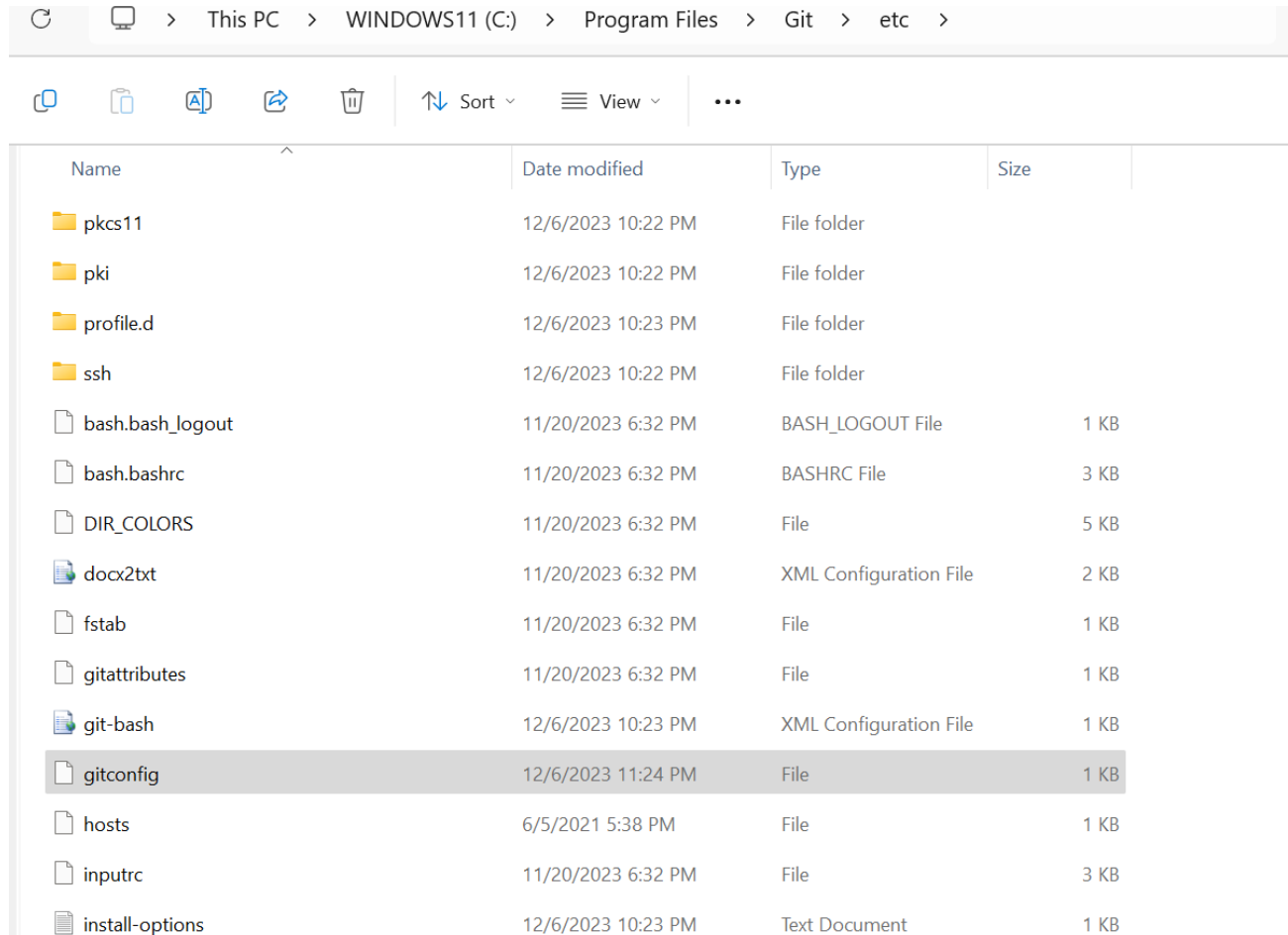
ELAIYE@ELAIYA MINGW64 ~ (master)
$ git config --system user.email "elaiyaraja_cs@sirmvit.edu"

ELAIYE@ELAIYA MINGW64 ~ (master)
$
```

The user name get overwrites as your emailid as user name.

# BCS358C-Project Management with Git

You can check in C:\Program Files\Git\etc

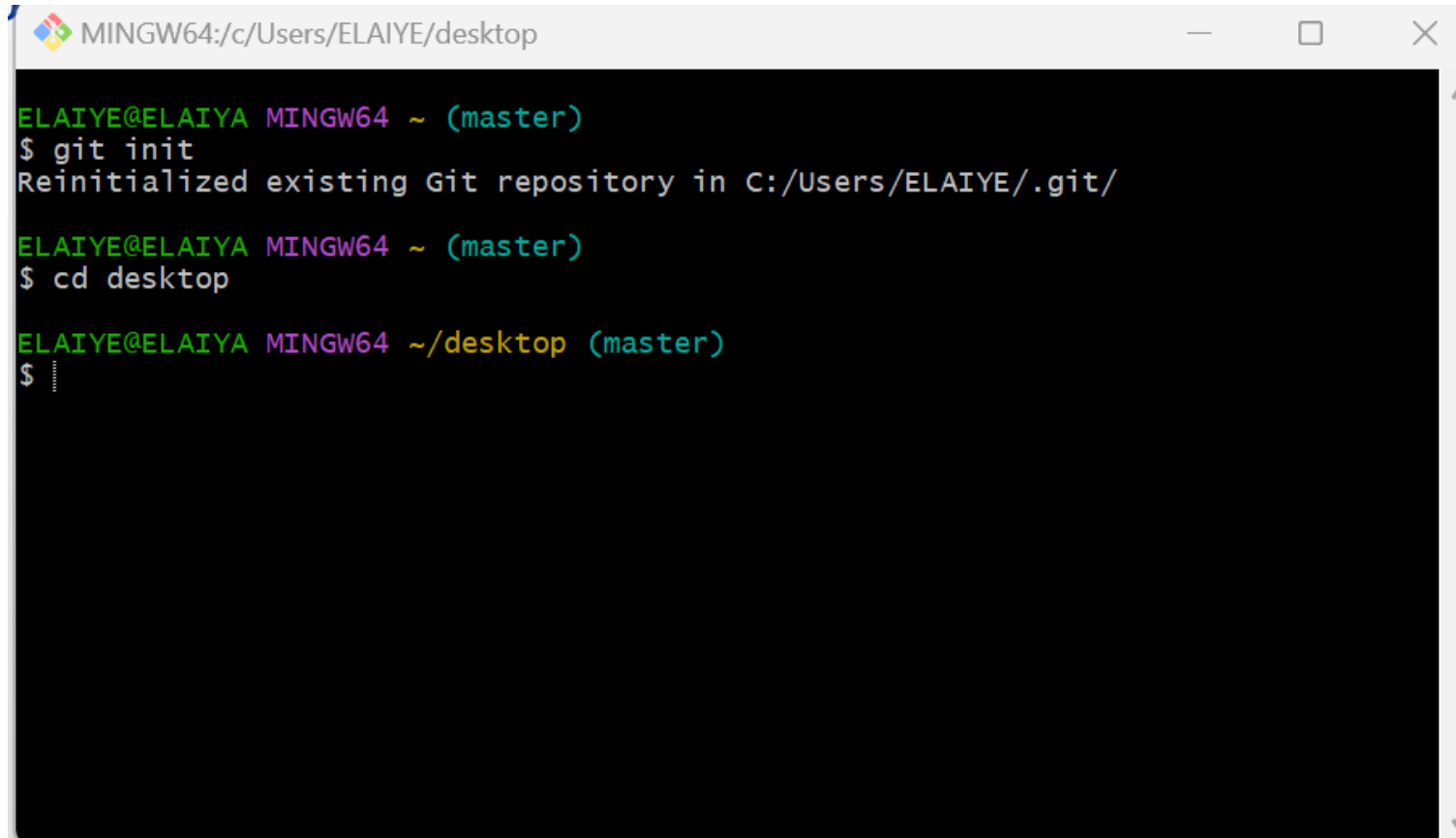


A screenshot of a Windows File Explorer window showing the directory C:\Program Files\Git\etc. The address bar at the top shows the path: This PC > WINDOWS11 (C:) > Program Files > Git > etc. The main area displays a list of files and folders. The 'gitconfig' file is highlighted in blue. The table below represents the data shown in the screenshot.

Name	Date modified	Type	Size
pkcs11	12/6/2023 10:22 PM	File folder	
pki	12/6/2023 10:22 PM	File folder	
profile.d	12/6/2023 10:23 PM	File folder	
ssh	12/6/2023 10:22 PM	File folder	
bash.bash_logout	11/20/2023 6:32 PM	BASH_LOGOUT File	1 KB
bash.bashrc	11/20/2023 6:32 PM	BASHRC File	3 KB
DIR_COLORS	11/20/2023 6:32 PM	File	5 KB
docx2txt	11/20/2023 6:32 PM	XML Configuration File	2 KB
fstab	11/20/2023 6:32 PM	File	1 KB
gitattributes	11/20/2023 6:32 PM	File	1 KB
git-bash	12/6/2023 10:23 PM	XML Configuration File	1 KB
gitconfig	12/6/2023 11:24 PM	File	1 KB
hosts	6/5/2021 5:38 PM	File	1 KB
inputrc	11/20/2023 6:32 PM	File	3 KB
install-options	12/6/2023 10:23 PM	Text Document	1 KB

Open gitconfig  
Using notepad once  
and you can able to  
view in the master  
User name

# BCS358C-Project Management with Git



```
MINGW64:/c/Users/ELAIYE/desktop
ELAIYE@ELAIYA MINGW64 ~ (master)
$ git init
Reinitialized existing Git repository in C:/Users/ELAIYE/.git/
ELAIYE@ELAIYA MINGW64 ~ (master)
$ cd desktop
ELAIYE@ELAIYA MINGW64 ~/desktop (master)
$
```

- Get into the  
desktop
- Cd desktop



# BCS358C-Project Management with Git

```
MINGW64:/c/Users/ELAIYE/desktop
ELAIYE@ELAIYA MINGW64 ~ (master)
$ git init
Reinitialized existing Git repository in C:/Users/ELAIYE/.git/

ELAIYE@ELAIYA MINGW64 ~ (master)
$ cd desktop

ELAIYE@ELAIYA MINGW64 ~/desktop (master)
$ mkdir gitlab

ELAIYE@ELAIYA MINGW64 ~/desktop (master)
$
```

Create directory

- `mkdir gitlab`

# BCS358C-Project Management with Git

```
MINGW64:/c/Users/ELAIYE/desktop/gitlab
ELAIYE@ELAIYA MINGW64 ~ (master)
$ git init
Reinitialized existing Git repository in C:/Users/ELAIYE/.git/

ELAIYE@ELAIYA MINGW64 ~ (master)
$ cd desktop

ELAIYE@ELAIYA MINGW64 ~/desktop (master)
$ mkdir gitlab

ELAIYE@ELAIYA MINGW64 ~/desktop (master)
$ cd gitlab

ELAIYE@ELAIYA MINGW64 ~/desktop/gitlab (master)
$ ls

ELAIYE@ELAIYA MINGW64 ~/desktop/gitlab (master)
$
```

Get into directory

- `cd gitlab`

# BCS358C-Project Management with Git

```
MINGW64:/c/Users/ELAIYE/desktop/gitlab
$ ls
ELAIYE@ELAIYA MINGW64 ~/desktop/gitlab (master)
$ ls
'gitlab class.txt'
ELAIYE@ELAIYA MINGW64 ~/desktop/gitlab (master)
$ git init
Initialized empty Git repository in C:/Users/ELAIYE/Desktop/gitlab/.git/
ELAIYE@ELAIYA MINGW64 ~/desktop/gitlab (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    gitlab class.txt

nothing added to commit but untracked files present (use "git add" to track)
ELAIYE@ELAIYA MINGW64 ~/desktop/gitlab (master)
$
```

Git can notify if your folder has some modified files

# BCS358C-Project Management with Git

```
MINGW64:/c/Users/ELAIYE/desktop/gitlab
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        gitlab class.txt
        gitlab class1.txt

nothing added to commit but untracked files present (use "git add" to track)

ELAIYE@ELAIYA MINGW64 ~/desktop/gitlab (master)
$ git add gitlab class.txt
fatal: pathspec 'gitlab' did not match any files

ELAIYE@ELAIYA MINGW64 ~/desktop/gitlab (master)
$ git add "gitlab class.txt"

ELAIYE@ELAIYA MINGW64 ~/desktop/gitlab (master)
$ git add "gitlab class1.txt"

ELAIYE@ELAIYA MINGW64 ~/desktop/gitlab (master)
$
```

Select which files to  
commit

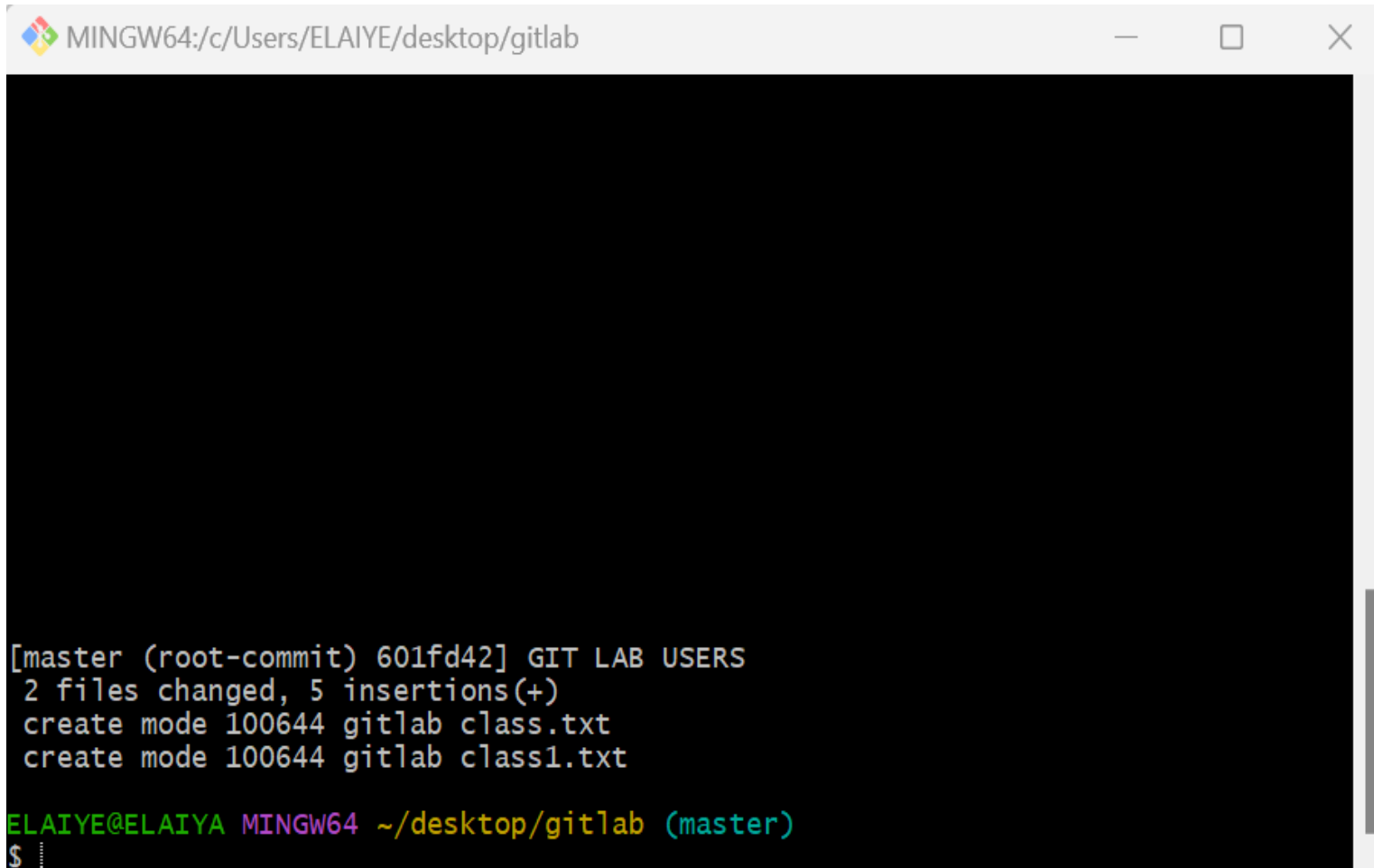
# BCS358C-Project Management with Git

```
MINGW64:/c/Users/ELAIYE/desktop/gitlab
```

```
# Please enter the commit message for your changes. Lines starting  
# with '#' will be ignored, and an empty message aborts the commit.  
#  
# On branch master  
#  
# Initial commit  
#  
# Changes to be committed:  
#   new file:   gitlab class.txt  
#   new file:   gitlab class1.txt  
#  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
~  
  
.git/COMMIT_EDITMSG [unix] (00:04 07/12/2023) 1,0-1 All  
"/Desktop/gitlab/.git/COMMIT_EDITMSG" [unix] 12L, 270B
```

- # Type
- git commit

# BCS358C-Project Management with Git



A screenshot of a terminal window titled 'MINGW64:/c/Users/ELAIYE/desktop/gitlab'. The terminal shows the output of a Git commit command. The output indicates that two files were changed with five insertions, and two new files, 'class.txt' and 'class1.txt', were created. The prompt at the bottom shows the user is in the 'master' branch of the 'gitlab' directory.

```
[master (root-commit) 601fd42] GIT LAB USERS
2 files changed, 5 insertions(+)
create mode 100644 gitlab class.txt
create mode 100644 gitlab class1.txt

ELAIYE@ELAIYA MINGW64 ~/desktop/gitlab (master)
$
```

Type  
Some content  
Press Esc key  
Type  
:wq to save.

# BCS358C-Project Management with Git

```
MINGW64:/c/Users/ELAIYE/desktop/gitlab
nothing to commit, working tree clean

ELAIYE@ELAIYA MINGW64 ~/desktop/gitlab (master)
$ commit
bash: commit: command not found

ELAIYE@ELAIYA MINGW64 ~/desktop/gitlab (master)
$ git commit
On branch master
nothing to commit, working tree clean

ELAIYE@ELAIYA MINGW64 ~/desktop/gitlab (master)
$ git diff

ELAIYE@ELAIYA MINGW64 ~/desktop/gitlab (master)
$ git log
commit 601fd4218982631d650530873c46d0b61e86a071 (HEAD -> master)
Author: gitlab <elaiyaraja_cs@sirmvit.edu>
Date: Thu Dec 7 00:04:40 2023 +0530

    GIT LAB USERS

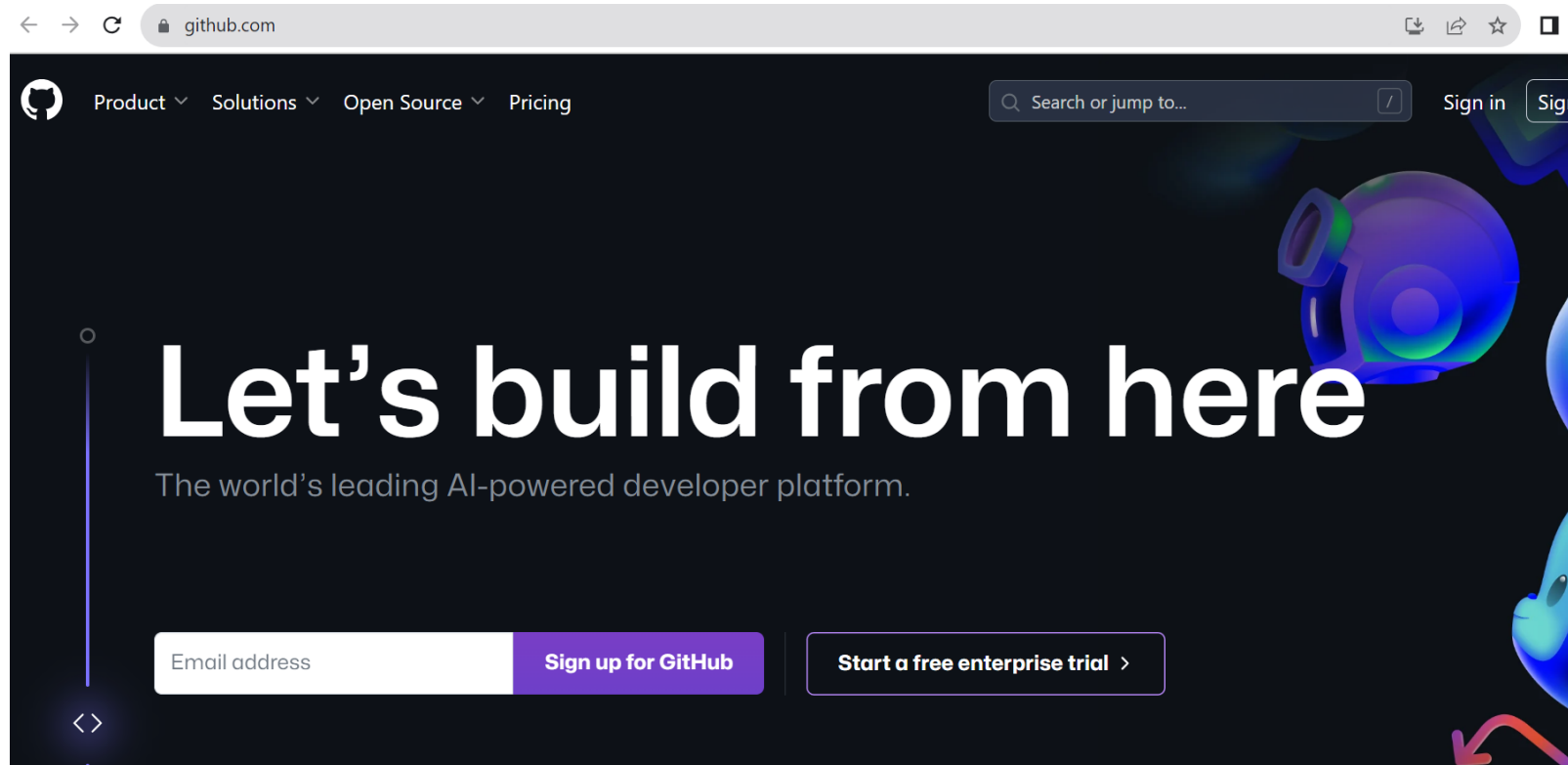
ELAIYE@ELAIYA MINGW64 ~/desktop/gitlab (master)
$
```

Type  
• **git log**  
to check your logs

# BCS358C-Project Management with Git

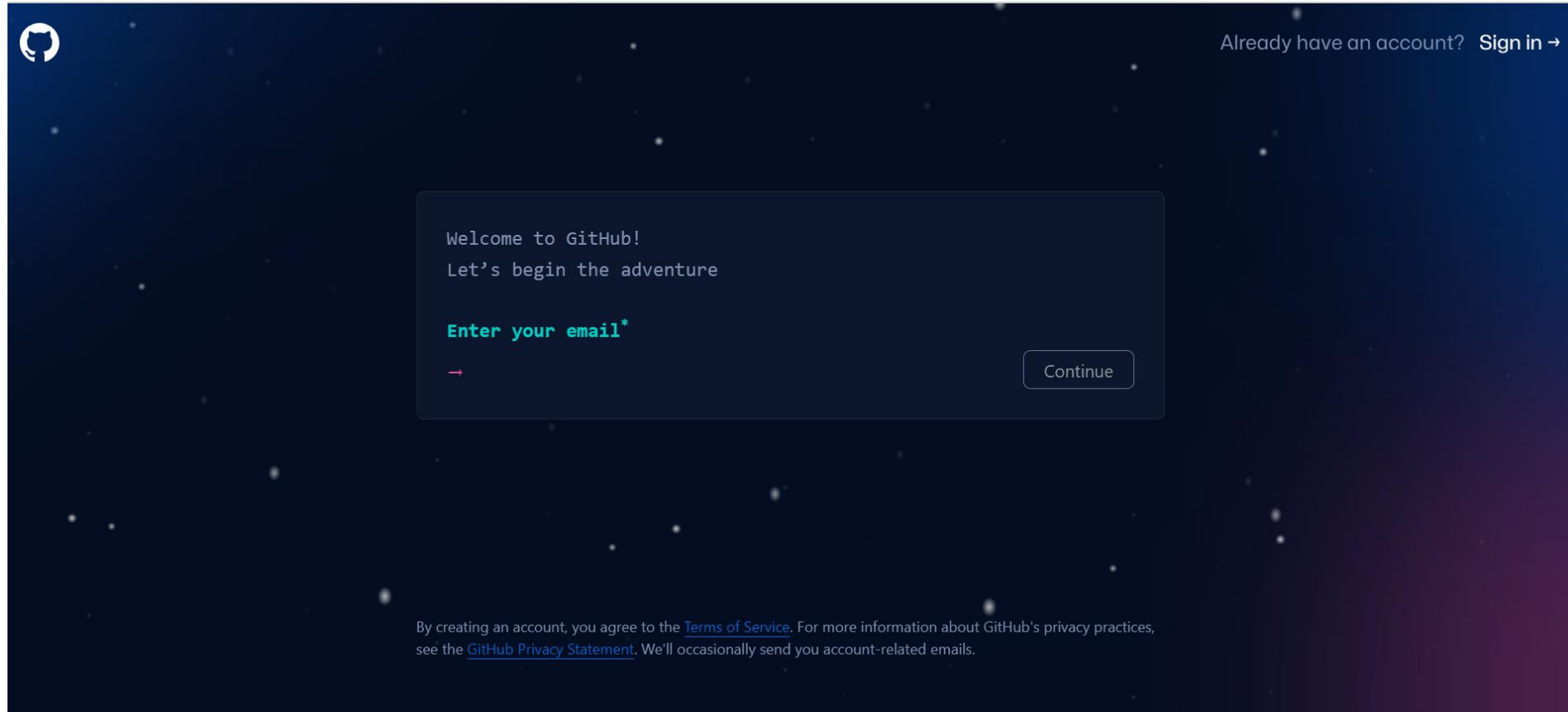
## How to push a repository to GitHub

### Step 1 – Create a GitHub account





# BCS358C-Project Management with Git



The image shows the GitHub sign-up page. At the top left is the GitHub logo. At the top right, it says "Already have an account? Sign in →". The main content area has a dark blue background with a starry pattern. A central white box contains the text "Welcome to GitHub! Let's begin the adventure". Below this is a text input field with the placeholder "Enter your email\*" and a red cursor. To the right of the input field is a "Continue" button. At the bottom, there is a small text block: "By creating an account, you agree to the [Terms of Service](#). For more information about GitHub's privacy practices, see the [GitHub Privacy Statement](#). We'll occasionally send you account-related emails."

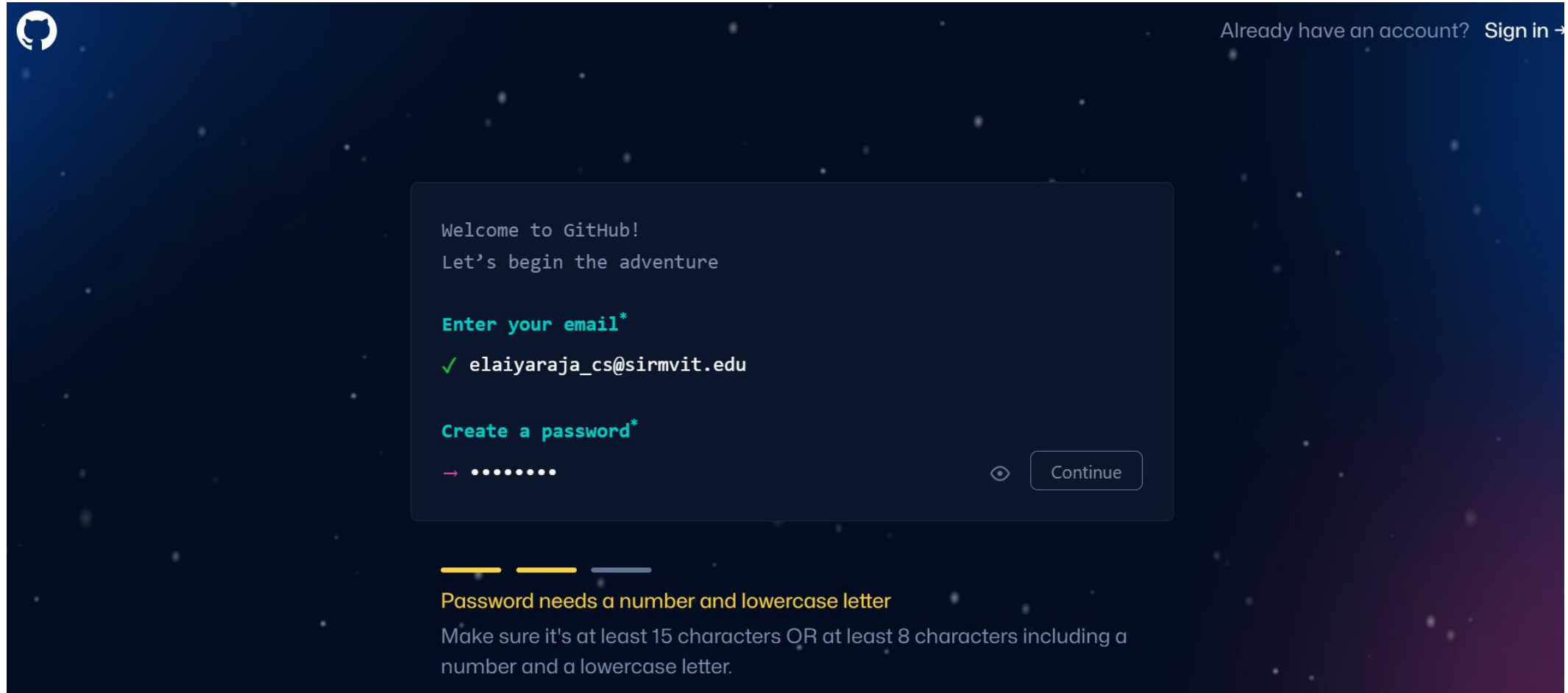
Welcome to GitHub!  
Let's begin the adventure

Enter your email\*

Continue

By creating an account, you agree to the [Terms of Service](#). For more information about GitHub's privacy practices, see the [GitHub Privacy Statement](#). We'll occasionally send you account-related emails.

# BCS358C-Project Management with Git



The image shows the GitHub sign-up interface. At the top left is the GitHub logo, and at the top right is the text "Already have an account? Sign in →". The main content area has a dark blue background with a starry pattern. A central white box contains the following text: "Welcome to GitHub!", "Let's begin the adventure", "Enter your email\*", "✓ elaiyaraja\_cs@sirmvit.edu", "Create a password\*", a password input field with a red arrow icon and a "Continue" button. Below this box, there are three colored bars (yellow, orange, blue) and a message: "Password needs a number and lowercase letter", followed by the instruction: "Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter."

Welcome to GitHub!  
Let's begin the adventure

Enter your email\*

✓ elaiyaraja\_cs@sirmvit.edu

Create a password\*

→ ••••••••

Continue

— — —

Password needs a number and lowercase letter

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter.

# BCS358C-Project Management with Git

The screenshot shows the GitHub Dashboard homepage. At the top is a navigation bar with the GitHub logo, the word "Dashboard", a search bar, and several utility icons. The main content area is divided into three columns. The left column contains a "Create your first project" section with a green "Create repository" button and a blue "Import repository" link, followed by a "Recent activity" section with a dashed box containing text about activity links. The middle column features a "Join GitHub Global Campus!" banner with a green "Join Global Campus" button, followed by a grid of various educational and career-related cards. The right column has a large "UNIVERSE23" banner for AI, Security, and DevEx, and a "Latest changes" section listing recent updates. At the bottom, there are links for "Home", "Send feedback", and a "Filter" button showing 8 items.

← → ↻ github.com

Dashboard

Search Type to search

Create your first project

Ready to start building? Create a repository for a new idea or bring over an existing repository to keep contributing to it.

Create repository Import repository

Recent activity

When you take actions across GitHub, we'll provide links to that activity here.

Join GitHub Global Campus!

Prepare for a career in tech by joining GitHub Global Campus. Global Campus will help you get the practical industry knowledge you need by giving you access to industry tools, events, learning resources and a growing student community.

Follow your Expert

Popular offers you have not claimed:

Curated Experiences with popular offers:

Virtual event kit

January 22, 2021

Level up your code with TwilioQuest

Ariel Kanter

February 1, 2021

GitHub Campus Experts applications are open

Juan Pablo Flores Cortés

Artificial Intelligence 6 assignments

Lists and Loops Due by May 1, 2021, 12:00 PST

Week Five: Functions Due by Aug 15, 2021, 14:00 PST

Connect your local Expert

View projects at our gallery

Learn more about an event

Watch a Campus TV episode

Claimed a Student Pack offer

Web meet 2021

Web Meet

Watch now

Latest changes

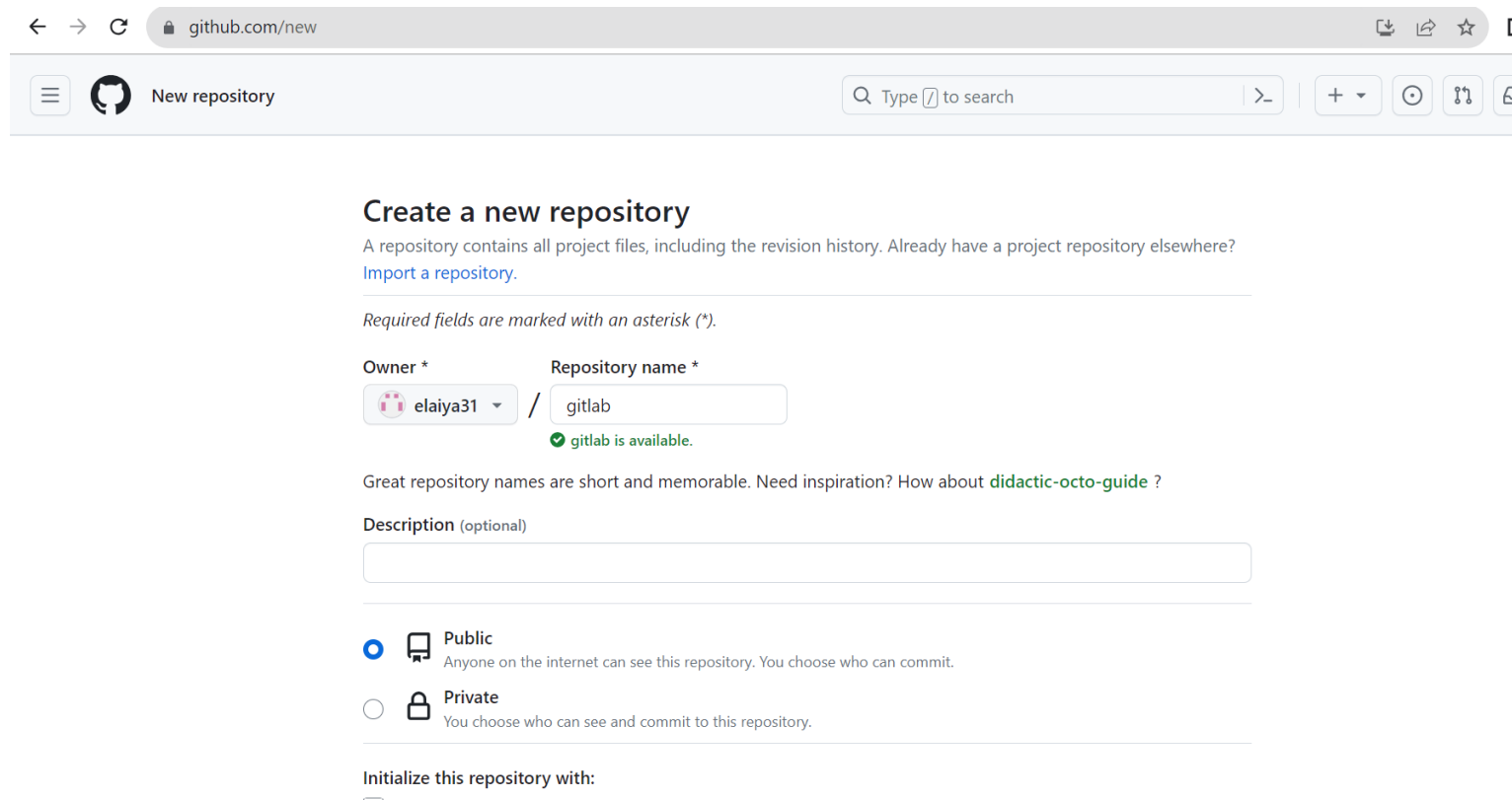
- 1 hour ago  
Updates to repository pages
- 7 hours ago  
New Organization Repositories List Feature Preview
- 19 hours ago  
GitHub Enterprise Server 3.11 is now generally available
- 2 days ago

Home Send feedback Filter 8

# BCS358C-Project Management with Git

## Step 2 – Create a repository

You can click on the + symbol on the top right corner of the page then choose "New repository". Give your repo a name then scroll down and click on "Create repository".



The screenshot shows the GitHub 'Create a new repository' page. The browser address bar shows 'github.com/new'. The page header includes the GitHub logo, 'New repository', a search bar, and navigation icons. The main content area is titled 'Create a new repository' and includes a description: 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)'. Below this, a note states 'Required fields are marked with an asterisk (\*)'. The 'Owner' field is set to 'elaiya31' and the 'Repository name' field is 'gitlab', with a green checkmark indicating 'gitlab is available'. A suggestion for 'didactic-octo-guide' is provided. The 'Description (optional)' field is empty. The 'Public' option is selected under the visibility settings. The 'Initialize this repository with:' section is partially visible at the bottom.

← → ↻ github.com/new


☰ GitHub New repository 🔍 Type / to search >\_ + ▾ ⌚ 🔗 📧

### Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

*Required fields are marked with an asterisk (\*).*


Owner \* Repository name \*


 elaiya31 / gitlab

✔ gitlab is available.

Great repository names are short and memorable. Need inspiration? How about [didactic-octo-guide](#) ?

Description (optional)

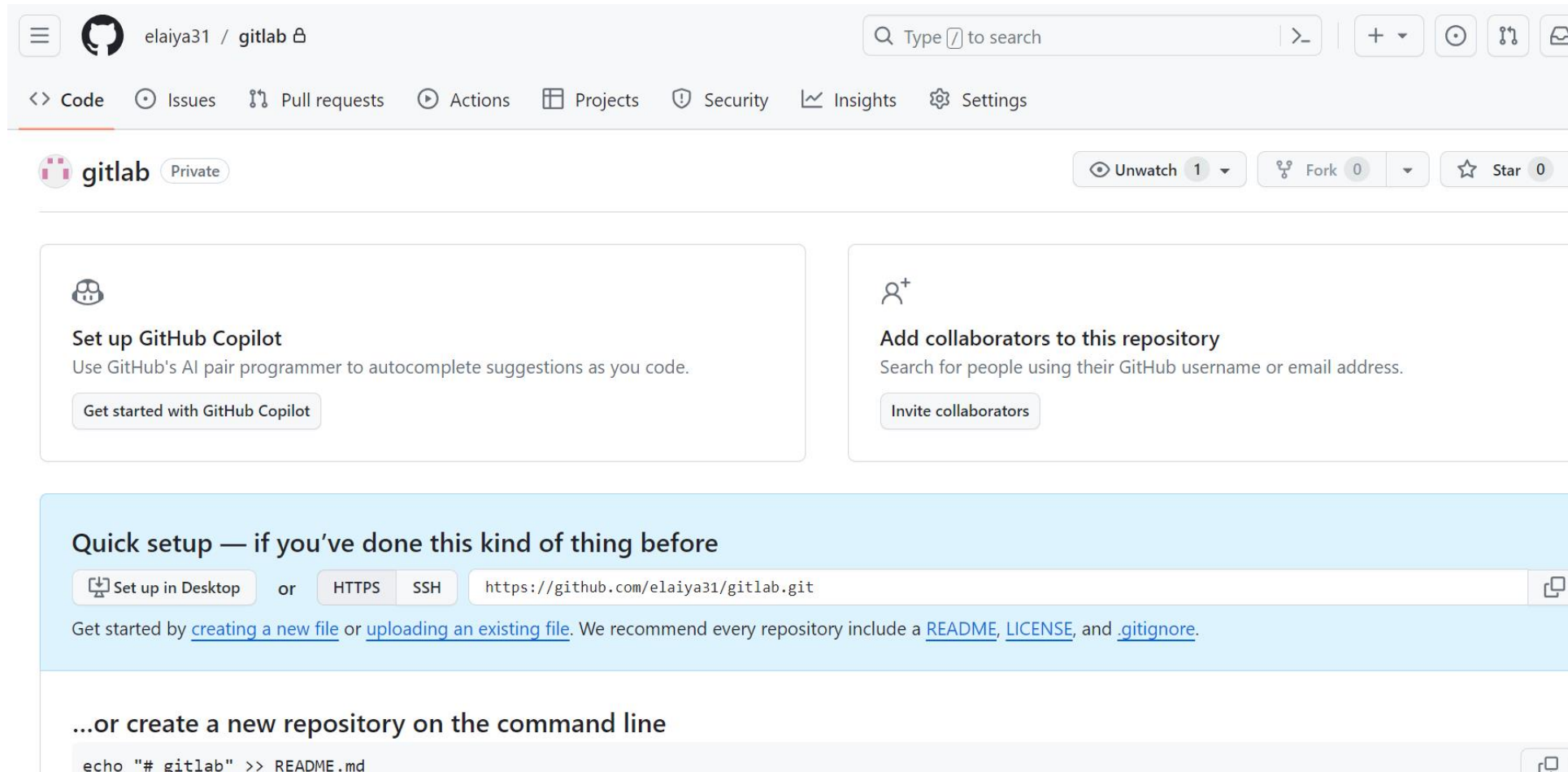
☒  **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**  
You choose who can see and commit to this repository.

Initialize this repository with:

# BCS358C-Project Management with Git

## Step 2 – Create a repository



The screenshot shows the GitHub interface for a new repository named 'gitlab' by user 'elaiya31'. The repository is private. The page offers options to set up GitHub Copilot, add collaborators, and provides a quick setup guide. The quick setup guide shows the repository URL as 'https://github.com/elaiya31/gitlab.git' and suggests including a README, LICENSE, and .gitignore. Below this, it shows the command to create a new repository on the command line: 'echo "# gitlab" >> README.md'.

elaiya31 / gitlab

Type / to search

Code Issues Pull requests Actions Projects Security Insights Settings

gitlab Private

Unwatch 1 Fork 0 Star 0

**Set up GitHub Copilot**  
Use GitHub's AI pair programmer to autocomplete suggestions as you code.  
[Get started with GitHub Copilot](#)

**Add collaborators to this repository**  
Search for people using their GitHub username or email address.  
[Invite collaborators](#)

**Quick setup — if you've done this kind of thing before**

[Set up in Desktop](#) or [HTTPS](#) [SSH](#) `https://github.com/elaiya31/gitlab.git`

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

**...or create a new repository on the command line**

```
echo "# gitlab" >> README.md
```

# BCS358C-Project Management with Git

## Basic commands

**git init:** Initializes a new Git repository in the current directory.

**git clone <repository URL>:** Creates a copy of a remote repository on your local machine.

**git add <file>:** Stages a file to be committed, marking it for tracking in the next commit.

**git commit -m "message":** Records the changes you've staged with a descriptive commit message.

**git status:** Shows the status of your working directory and the files that have been modified or staged.

## BCS358C-Project Management with Git

**git log:** Displays a log of all previous commits, including commit hashes, authors, dates, and commit messages.

**git diff:** Shows the differences between the working directory and the last committed version.

**git branch:** Lists all branches in the repository and highlights the currently checked-out branch.

**git branch <branchname>:** Creates a new branch with the specified name.

**git checkout <branchname>:** Switches to a different branch.

**git merge <branchname>:** Merges changes from the specified branch into the currently checked-out branch.

# BCS358C-Project Management with Git

## **EXPERIMENT 1 :**

### **Setting Up and Basic Commands**

**Initialize a new Git repository in a directory. Create a new file and add it to the staging area and commit the changes with an appropriate commit message.**



# BCS358C-Project Management with Git

**\$ git init**

1. Create a new file in the directory. For example, let's create a file named "my\_file.txt."  
You can use any text editor or command-line tools to create the file.
2. Add the newly created file to the staging area. Replace "my\_file.txt" with the actual name of your file:

**\$ git add my\_file.txt**

This command stages the file for the upcoming commit.

# BCS358C-Project Management with Git

```
$ git commit -m "Your commit message here"
```

Your commit message should briefly describe the purpose or nature of the changes you made. For example:

```
$ git commit -m "Add a new file called my_file.txt"
```

After these steps, your changes will be committed to the Git repository with the provided commit message.

You now have a version of the repository with the new file and its history stored in Git.

# BCS358C-Project Management with Git

## **EXPERIMENT 2 :**

### **Creating and Managing Branches**

**Create a new branch named "feature-branch." Switch to the  
"master" branch. Merge the  
"feature-branch" into "master."**

# BCS358C-Project Management with Git

## Creating and Managing Branches:

Create a new branch named "feature branch." Switch to the "master" branch. Merge the "feature-branch" into "master."

### Solution:

To create a new branch named "feature-branch," switch to the "master" branch, and merge the "feature-branch" into "master" in Git, follow these steps:

1. Make sure you are in the "master" branch by switching to it:

**\$ git checkout master**

# BCS358C-Project Management with Git

1. Create a new branch named "feature-branch" and switch to it:

**\$ git checkout -b feature-branch**

This command will create a new branch called "feature-branch" and switch to it.

1. Make your changes in the "feature-branch" by adding, modifying, or deleting files as needed.
2. Stage and commit your changes in the "feature-branch":

# BCS358C-Project Management with Git

git add .

**\$ git commit -m "Your commit message for feature-branch"**

Replace "Your commit message for feature-branch" with a descriptive commit message for the changes you made in the "feature-branch."

1. Switch back to the "master" branch:

**\$ git checkout master**

# BCS358C-Project Management with Git

1. Merge the "feature-branch" into the "master" branch:

**\$ git merge feature-branch**

This command will incorporate the changes from the "feature-branch" into the "master" branch.

Now, your changes from the "feature-branch" have been merged into the "master" branch. Your project's history will reflect the changes made in both branches

# BCS358C-Project Management with Git

## **EXPERIMENT 3 :**

### **Creating and Managing Branches**

**Write the commands to stash your changes, switch branches, and then apply the stashed changes.**



# BCS358C-Project Management with Git

## Creating and Managing Branches:

Write the commands to stash your changes, switch branches, and then apply the stashed changes.

### Solution:

To stash your changes, switch branches, and then apply the stashed changes in Git, you can use the following commands:

1. Stash your changes:

**\$ git stash save "Your stash message"**

This command will save your changes in a stash, which acts like a temporary storage for changes that are not ready to be committed.

# BCS358C-Project Management with Git

1. Switch to the desired branch:

**\$ git checkout target-branch**

Replace "target-branch" with the name of the branch you want to switch to.

1. Apply the stashed changes:

**\$ git stash apply**

This command will apply the most recent stash to your current working branch. If you have multiple stashes, you can specify a stash by name or reference (e.g., `git stash apply stash@{2}`) if needed.

If you want to remove the stash after applying it, you can use `git stash pop` instead of `git stash apply`.

Remember to replace "Your stash message" and "target-branch" with the actual message you want for your stash and the name of the branch you want to switch to.

# BCS358C-Project Management with Git

## **EXPERIMENT 4 :**

**Collaboration and Remote Repositories:**

**Clone a remote Git repository to your local machine.**

# BCS358C-Project Management with Git

1. Open your terminal or command prompt.
2. Navigate to the directory where you want to clone the remote Git repository. You can use the `cd` command to change your working directory.
3. Use the `git clone` command to clone the remote repository. Replace `<repository_url>` with the URL of the remote Git repository you want to clone. For example, if you were cloning a repository from GitHub, the URL might look like this:

**\$ git clone <repository\_url>**

**\$ git clone <https://github.com/username/repo-name.git>**

**Ex: <https://github.com/elaiya31/3githubnotes>**

# BCS358C-Project Management with Git

1. Git will clone the repository to your local machine. Once the process is complete, you will have a local copy of the remote repository in your chosen directory.

You can now work with the cloned repository on your local machine, make changes, and push those changes back to the remote repository as needed.

# BCS358C-Project Management with Git

## **EXPERIMENT 5 :**

### **Collaboration and Remote Repositories:**

**Fetch the latest changes from a remote repository and rebase your local branch onto the updated remote branch.**

# BCS358C-Project Management with Git

1. Open your terminal or command prompt.
2. Make sure you are in the local branch that you want to rebase. You can switch to the branch using the following command, replacing `<branch-name>` with your actual branch name:

**\$ git checkout <branch-name>**

# BCS358C-Project Management with Git

1. Fetch the latest changes from the remote repository. This will update your local repository with the changes from the remote without merging them into your local branch:

**\$ git fetch origin**

Here, origin is the default name for the remote repository. If you have multiple remotes, replace origin with the name of the specific remote you want to fetch from.



# BCS358C-Project Management with Git

1. Once you have fetched the latest changes, rebase your local branch onto the updated remote branch:

**\$ git rebase origin/<branch-name>**

Replace <branch-name> with the name of the remote branch you want to rebase onto. This command will reapply your local commits on top of the latest changes from the remote branch, effectively incorporating the remote changes into your branch history.

## BCS358C-Project Management with Git

1. Resolve any conflicts that may arise during the rebase process. Git will stop and notify you if there are conflicts that need to be resolved. Use a text editor to edit the conflicting files, save the changes, and then continue the rebase with:

**\$ git rebase --continue**

# BCS358C-Project Management with Git

1. After resolving any conflicts and completing the rebase, you have successfully updated your local branch with the latest changes from the remote branch.
2. If you want to push your rebased changes to the remote repository, use the git push command. However, be cautious when pushing to a shared remote branch, as it can potentially overwrite other developers' changes:

**\$ git push origin <branch-name>**

Replace <branch-name> with the name of your local branch. By following these steps, you can keep your local branch up to date with the latest changes from the remote repository and maintain a clean and linear history through rebasing.

# BCS358C-Project Management with Git

## **EXPERIMENT 6**

### **Collaboration and Remote Repositories:**

**Write the command to merge "feature-branch" into "master" while providing a custom commit message for the merge.**

# BCS358C-Project Management with Git

To merge the "feature-branch" into "master" in Git while providing a custom commit message for the merge, you can use the following command:

```
$ git checkout master
```

```
$ git merge feature-branch -m "Your custom commit message here"
```

Replace "Your custom commit message here" with a meaningful and descriptive commit message for the merge. This message will be associated with the merge commit that is created when you merge "feature-branch" into "master."

# BCS358C-Project Management with Git

## **EXPERIMENT 7.**

### **Git Tags and Releases:**

**Write the command to create a lightweight Git tag named "v1.0" for a commit in your local repository.**

# BCS358C-Project Management with Git

To create a lightweight Git tag named "v1.0" for a commit in your local repository, you can use the following command:

```
$ git tag v1.0
```

This command will create a lightweight tag called "v1.0" for the most recent commit in your current branch. If you want to tag a specific commit other than the most recent one, you can specify the commit's SHA-1 hash after the tag name. For example:

```
$ git tag v1.0 <commit-SHA>
```

Replace <commit-SHA> with the actual SHA-1 hash of the commit you want to tag.

# BCS358C-Project Management with Git

## **Experiment 8.**

### **Advanced Git Operations:**

**Write the command to cherry-pick a range of commits  
from "source-branch" to the current branch.**



# BCS358C-Project Management with Git

To cherry-pick a range of commits from "source-branch" to the current branch, you can use the following command:

```
$git cherry-pick <start-commit>^..<end-commit>
```

Replace <start-commit> with the commit at the beginning of the range, and <end-commit> with the commit at the end of the range. The ^ symbol is used to exclude the <start-commit> itself and include all commits after it up to and including <end-commit>. This will apply the changes from the specified range of commits to your current branch. For example, if you want to cherry-pick a range of commits from "source-branch" starting from commit ABC123 and ending at commit DEF456, you would use:

```
$ git cherry-pick ABC123^..DEF456
```

Make sure you are on the branch where you want to apply these changes before running the cherry-pick command.

# BCS358C-Project Management with Git

## **Experiment 9.**

### **Analysing and Changing Git History:**

**Given a commit ID, how would you use Git to view the details of that specific commit, including the author, date, and commit message?**

# BCS358C-Project Management with Git

To view the details of a specific commit, including the author, date, and commit message, you can use the `git show` or `git log` command with the commit ID. Here are both options:

1. Using `git show`:

```
bash
```

```
git show <commit-ID>
```

Replace `<commit-ID>` with the actual commit ID you want to view. This command will display detailed information about the specified commit, including the commit message, author, date, and the changes introduced by that commit.

# BCS358C-Project Management with Git

For example:

```
$ git show abc123
```

1. Using git log:

```
$ git log -n 1 <commit-ID>
```

The -n 1 option tells Git to show only one commit. Replace <commit-ID> with the actual commit ID. This command will display a condensed view of the specified commit, including its commit message, author, date, and commit ID.

# BCS358C-Project Management with Git

For example:

```
$ git log -n 1 abc123
```

Both of these commands will provide you with the necessary information about the specific commit you're interested in.

# BCS358C-Project Management with Git

## **Experiment 10.**

### **Analysing and Changing Git History:**

**Write the command to list all commits made by the author**

**"JohnDoe" between "2023- 01-01"and "2023-12-31."**

## BCS358C-Project Management with Git

To list all commits made by the author "JohnDoe" between "2023-01-01" and "2023-12-31" in Git, you can use the git log command with the --author and --since and --until options. Here's the command:

```
$ git log --author="JohnDoe" --since="2023-01-01" --until="2023-12-31"
```

This command will display a list of commits made by the author "JohnDoe" that fall within the specified date range, from January 1, 2023, to December 31, 2023. Make sure to adjust the author name and date range as needed for your specific use case.

# BCS358C-Project Management with Git

## **Experiment 11.**

### **Analysing and Changing Git History:**

**Write the command to display the last five commits in the repository's history.**



## BCS358C-Project Management with Git

To display the last five commits in a Git repository's history, you can use the git log command with the -n option, which limits the number of displayed commits. Here's the command:

```
$ git log -n 5
```

This command will show the last five commits in the repository's history. You can adjust the number after -n to display a different number of commits if needed.

# BCS358C-Project Management with Git

## **Experiment 12.**

### **Analysing and Changing Git History:**

**Write the command to undo the changes introduced by the commit with the ID "abc123".**

## BCS358C-Project Management with Git

To undo the changes introduced by a specific commit with the ID "abc123" in Git, you can use the `git revert` command. The `git revert` command creates a new commit that undoes the changes made by the specified commit, effectively "reverting" the commit.

Here's the command:

```
$ git revert abc123
```

Replace "abc123" with the actual commit ID that you want to revert. After running this command, Git will create a new commit that negates the changes introduced by the specified commit. This is a safe way to undo changes in Git because it preserves the commit history and creates a new commit to record the reversal of the changes.

# BCS358C-Project Management with Git

**TEST : LAB INTERNAL - I**

**Subject Code : BCS358C**

**Subject Name : Project Management with Git**

1	3 <sup>RD</sup> SEMESTER- A SEC.	B1	05/03/2024	9.30 TO 11.30 AM
2		B2	05/03/2024	1.35 TO 3.35 PM

# BCS358C-Project Management with Git

**THE END**